AD-A071 334 AIR FORCE AERO PROPULSION LAB WRIGHT-PATTERSON AFB OH F/G 9/2
AN IMPROVED FORTRAN REORGANIZER.(U)
MAY 79 K D MACH
UNCLASSIFIED AFAPL-TR-79-2052 NL

1 OF 2
AD
A071334

LEVEL II

2

B. S.

AFAPL-TR-79-2052

AD A071334

AN IMPROVED FORTRAN REORGANIZER

TURBINE ENGINE DIVISION
COMPONENTS BRANCH

May 1979

DDC FILE COPY

TECHNICAL REPORT AFAPL-TR-79-2025
Final Report for Period August 1977 to February 1979

AIR FORCE AERO-PROPULSION LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

79 07 17 043

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Kervyn D Mach

KERVYN D. MACH
Project Engineer

[illegible]

JOSEPH C. HURST, Major, USAF
Chief, Components Branch

FOR THE COMMANDER

[illegible]

ERNEST C. SIMPSON
Director
Turbine Engine Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAPL/TBC *,W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		**READ INSTRUCTIONS BEFORE COMPLETING FORM**
(14) 1. REPORT NUMBER AFAPL-TR-79-2052	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER rept.
(6) 4. TITLE (and Subtitle) An Improved Fortran Reorganizer		(9) 5. TYPE OF REPORT & PERIOD COVERED Interim Aug 77 ~~to~~ Feb 79 ~~6. PERFORMING ORG. REPORT NUMBER~~
(10) 7. AUTHOR(s) Kervyn D. Mach		8. CONTRACT OR GRANT NUMBER(s) In-House
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFAPL/TBC Wright-Patterson AFB OH 45433		(16) (17) 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS P.E. 62203F, Proj 3066, Task 06, Work Unit 02
11. CONTROLLING OFFICE NAME AND ADDRESS AFAPL/TBC Wright-Patterson AFB OH 45433		(11) 12. REPORT DATE May 1979 ~~13. NUMBER OF PAGES~~ 91
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 97p.		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer programs and programming, Fortran

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Because this office frequently distributes Fortran computer programs to other organizations, it needed a means to furnish these programs in a clean, easy to read format. This report describes a computer program designed to provide such a format by reorganizing Fortran routines. This reorganization includes: a sequential renumbering of the executable statement label with deletion of unused labels; a sequential renumbering and relocation of format statements; an alphanumeric reordering of dimensional and typed variables; and a uniform pattern of text spacing. The program has been extensively tested

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

011 570 LB

20.
and has proved to be extremely valuable for reorganizing Fortran routines developed in-house and under contract and for preparing routines for distribution and publication.

FOREWORD

This report describes work conducted within the Air Force Aero-Propulsion Laboratory, Turbine Engine Division, Components Branch (TBC), Wright-Patterson Air Force Base, Ohio. The work was accomplished under Project 3066, "Gas Turbine Technology," Task 06, "Turbine Technology," Work Unit 02, "Turbine Aeromechanical Analysis," between August 1977 and February 1979.

The report was submitted by the author in May 1979.

TABLE OF CONTENTS

1. INTRODUCTION

The work of the Turbine Components Branch of the AF Aero Propulsion Laboratory includes the development and purchase of numerous digital computer programs coded in FORTRAN. Many of these programs are subsequently distributed to other organizations, both within and outside the Government.

Maintenance of a large inventory of programs requires that they be documented in a consistent, easy-to-understand format. This also simplifies the implementation task for those people to whom we distribute programs. Accordingly, we sought a means, preferably another computer program, which would accept a working FORTRAN program as input and deliver a reformatted equivalent as output. As a minimum, we wanted the statement labels resequenced in increasing order (e.g., 1, 2, 3, ...) and the program to handle any statement accepted by the CDC FORTRAN Extended compiler.

As our first attempt in this direction, we acquired TIDY and used it for about two years. TIDY, however, was not completely satisfactory. Although it resequenced the statement labels, it also deleted almost all blanks from the reformatted statements which made the resultant listing somewhat difficult to follow. More seriously, it did not recognize some of the statement forms accepted by the CDC FORTRAN Extended compiler. For instance, the input record

```
READ *, A $ B = SQRT(A) $ IF(A.GT.3.0) GO TO 383
```

would not be properly handled because TIDY does not recognize the list-directed read command nor does it recognize the dollar sign as a statement separator. Consequently, it does not process the logical IF at all. To be properly processed by TIDY, the sequence above would have to be written

```
      READ 100, A
  100 FORMAT (F10.0)
      B = SQRT(A)
      IF (A.GT.3.0) GO TO 383
```

i.e., we would have to program in a subset of FORTRAN Extended if we wished to use TIDY.

In 1977 we acquired REOR, which forms the basis for the program described in this report. REOR was coded in FORTRAN Extended and specifically recognized the dollar sign statement separators (it outputs such statements on separate lines) and offered a somewhat more readable output format. Though it also did not recognize certain FORTRAN Extended dialect statements, REOR proved to be very easy to modify to achieve the capability we wanted. The remainder of this report briefly describes the features of REOR and describes in more detail the added features of the augmented program, known as CLEAN. Appendix A contains a complete listing of the program and Appendix B contains descriptions of subroutines which were added or significantly changed.

2. THE PROGRAM

2.1 Original Features

As acquired, REOR provided the following major capabilities:

a. Labels on executable statements were renumbered in sequence, e.g., 1000, 1010, 1020, etc.

b. FORMAT statements were gathered at the end of the program module and renumbered 10, 20, 30, etc. Unreferenced FORMAT statements were deleted.

c. Variables in DIMENSION and type statements were gathered into single statements of the appropriate type and output in alphanumeric order.

d. DO loops were indented two spaces each.

e. Blank spaces were added for readability, e.g., two on either side of an equals sign, one before each left parenthesis, one after each right parenthesis, one after each comma, etc.

2.2 Added Features

We made a number of additions and modifications to REOR in order to achieve the capabilities and output format we wanted:

a. It was converted to overlay form to fit within the memory limits of the local implementation of CDC's INTERCOM interactive system. The program can now be used interactively or in batch mode.

b. The program output is to a file called TFILE, which is rewound at the end of the job. The contents of TFILE may be punched, listed, compiled, or whatever.

c. It recognizes and properly processes some additional FORTRAN statements:

(1) OVERLAY, as in OVERLAY (FILE, m, n)

(2) IMPLICIT, as in IMPLICIT REAL (I - K)

(3) COMMON//, as in COMMON// A, B, C

(4) List - directed input - output, as in

```
READ *, list
READ (5,*) list
PRINT *, list
WRITE (6,*) list
```

d. Unreferenced labels on executable statements are deleted, as explained in Appendix B. The remainder are renumbered 1, 2, 3, ... FORMAT statements are renumbered 100, 101, 102, ..., unless there are more than 99 executable labels. In that case, the program tries 200, then 300, etc., until it finds a number larger than the last executable label.

e. The program recognizes and properly handles Hollerith literals in any statement in which they are legal, viz:

```
    J = 6HSTRING
    DATA JR /6RSTRING/
    CALL CONNEC (6LOUTPUT)
    IF (J. EQ. "STRING")...
100 FORMAT (6., *STR*, 'ING')
```

In the last case, 'ING' is changed to "ING".

f. Nonstandard returns from subroutines are detected and the new labels inserted, as in CALL TAXI (A, J, "DOWNTOWN"), RETURNS (40,58).

g. The output format has been slightly modified to suit our preference. Blanks are not inserted before and after parentheses in replacement statements. One blank follows each equals sign and at least one blank precedes. If the equals sign lies to the left of column

18, it is moved to column 18 and the vacated space is filled with blanks. (In DO loops, the entire statement is moved right after the equals sign is positioned.) The original REOR put blanks before and after each arithmetic operator (* / + -), which left two blanks in the middle of the exponentiation operator, so that ** became * *. The revised program puts blanks before the first and after the second asterisk, but not between.

The revised program has been renamed CLEAN. The listing in Appendix A gives a good example of its output format.

2.3 Limitations

Program CLEAN will accept and process any legal FORTRAN Extended statement except references to extended core storage (ECS). This omission is unimportant since our computer does not have ECS.

Conversion to other compilers would entail the same difficulties as for the original REOR. The program assumes ten characters per word storage and makes frequent use of the ENCODE and DECODE instructions, which are not available on all compilers.

2.4 Execution Speed

CLEAN will process approximately 100 source cards per second of CDC 6600 central processor time.

References

1. Marvin S. Seppanen, A Fortran Routine Reorganizer, BuMines IC 8696, Twin Cities Mining Research Center, Bureau of Mines, USDI, Twin Cities, Minnesota, January 1976.

2. FORTRAN Extended Version 4 Reference Manual, Publication No. 60497800, Control Data Corporation, Sunnyvale, California, November 1975.

3. Alice V. Barlow and Gary N. Vanderplaats, Tidy, A Complete Code for Renumbering and Editing Fortran Source Programs, NASA TM X-62886, Computer Sciences Corporation, Mountain View, California, and Ames Research Center and US Army Air Mobility R&D Laboratory, Moffett Field, California, August 1973.

APPENDIX A

Listing of CLEAN

The complete source code of CLEAN is presented herein. The listings show an occasional continuation line marked with a dollar sign. These were created by the listing program to maintain the right-hand margin and do not appear in the actual code.

```
      OVERLAY(CLEAN,0,0)
      PROGRAM CLEAN (INPUT, OUTPUT, TFILE, TAPE3, TAPE4=
     $ TFILE,
     1  TAPE2=INPUT)
C
C       THIS PROGRAM READS A STANDARD FORTRAN ROUTINE
C    $          FILE AND
C            REORGANIZES THE ROUTINE BY ORDERING THE
C    $          STATEMENT NUMBERS
C         AND ADJUSTING THE STATEMENT SPACING AND SEQUENCE.
C
      COMMON  /ALL/  ICHARS, IBOLLER, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4  NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5  (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IFUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3  STAR, X
      COMMON  /SNLIST/  NS, REF (400, 3)
      INTEGER      C, END, H, IDATA (4613), PROGRAM, REF,
     $ RETURN, STAR,
     1  STRING, X
      EQUIVALENCE  (ICHARS, IDATA(1))
      DATA     ICOUNT, IDATA / 8 * 0, 4613 * 0 /
      DATA     C, END, H, IBLANK, IEOF / 1HC, 3HEND, 1HH,
     $ 1H , 0 /
      DATA     INTEGER / 1H0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6,
     $ 1H7, 1H8,
     1  1H9 /
      DATA     IFUNCT / 1H/, 1H,, 1H(, 1H), 1H*, 1H$, 1H.,
     $ 1H=, 1H-,
     1  1H+, 1H" /
      DATA     LUIN, LUOUT, LUSTATE, MFOUT, MLCHARS,
     $ MNFORM, MNSTATE,
     1  NCARD, NMAX, NUMMAX, RETURN, STAR, X / 2, 4, 3,
     $ 1000, 2000, 99,
     2  400, 0, 100, 50, 6HRETURN, 1H*, 1HX /
      DATA     PROGRAM / 1HP, 1HR, 1HO, 1HG, 1HR, 1HA, 1HM
     $ /
```

PRECEDING PAGE BLANK-NOT FILMED

```
C
C        HOUSEKEEPING
C
    1  CALL RESETS
C
C        DO THE READ CYCLE.  READ THE STATEMENTS FROM THE
C     $        INPUT FILE
C        TAPE2, PROCESS, AND STORE ON THE WORKING FILE
C     $        TAPE3.
C
       CALL OVERLAY (5HCLEAN, 1, 0)
C
C        DO THE WRITE CYCLE.  READ THE STATEMENTS FROM THE
C     $        WORKING
C        FILE, COMPLETE THE PROCESSING, AND WRITE TO
C     $        TAPE4.
C
       CALL OVERLAY (5HCLEAN, 2, 0)
C
C        REPEAT IF NO EOF ENCOUNTERED.
C
       IF (IEOF .EQ. 0) GO TO 1
       REWIND 4
       STOP
       END
```

```
      SUBROUTINE  ALIGN
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IFUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
C
      INTEGER       COMMAS
      COMMAS      = 0
      IP          = 1
C
C        CHECK FOR THE EQUAL SIGN.  PUT IT IN COL 18 IF
C    $         POSSIBLE.
C        ALWAYS PUT A BLANK FOLLOWING AND AT LEAST ONE
C    $         PRECEDING.
C
      IPOINT      = ISCANL(IFUNCT(8), IP, LCHARS, LSTATE(1))
        DO 1 I      = 1, IPOINT
        IF  (LSTATE(I) .EQ. IFUNCT(2)) COMMAS = COMMAS + 1
    1   CONTINUE
      IF (IPOINT .LT. LCHARS) GO TO 3
      PRINT  100, (LSTATE(I), I=1, LCHARS)
      RETURN
    2 IPOINT      = ISCANL(IFUNCT(8), IP, LCHARS, LSTATE(1))
      IF (IPOINT .GE. LCHARS) RETURN
C
C        FOLLOWING
C
    3 CALL INSERT (IBLANK, IPOINT + 1, LCHARS, LSTATE(1),
     $ 1)
C
C        PRECEDING
C
      MANY        = 1
      IF  (IPOINT .LT. 11 + I9999) MANY = MAX0 (1,11 +
     $ I9999 - IPOINT -
     1   COMMAS)
         CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1),
        $ MANY)
C
C            LOOK FOR MULTIPLE REPLACEMENTS
C
         IP          = IPOINT + MANY + 2
         COMMAS      = 0
         GO TO 2
C
  100    FORMAT  (  *0COULD NOT FIND AN EQUAL SIGN IN THIS*,
        $ * REPLACEME*
     1     *NT STATEMENT.* / (1X, 130A1) )
C
         END
```

```
      LOGICALFUNCTION  CHECK (LOOK4, NN, ISTART, ISTOP,
     $ LIST, IPOINT)
C
C     THIS FUNCTION SCANS THE STRING 'LIST' FROM ISTART TO
C     ISTOP FOR THE SPECIFIC STRING 'LOOK4'. BLANKS IN
C    $           'LIST'
C      ARE IGNORED.
C
      DIMENSION    LIST (1), LOOKUP  (10)
      DATA     IBLANK / 1H  /
C     IF (ISTOP - ISTART .LT. NN)
      IF ( .NOT. (ISTOP-ISTART .LT. NN)) GO TO 1
      CHECK        = .FALSE.
      IPOINT       = ISTART
      RETURN
C     END IF
    1 CONTINUE
      DECODE (10,  100, LOOK4)  LOOKUP
C     DO (IPOINT = ISTART,ISTOP - NN + 1)
      I99996       = ISTART
      I99995       = ISTOP  -  NN + 1
        DO 10 IPOINT = I99996, I99995
C
C     FIND A POSSIBLE START POINT.
C
C     IF (LIST(IPOINT) .EQ. LOOKUP(1))
        IF ( .NOT. (LIST(IPOINT) .EQ. LOOKUP(1))) GO TO 9
        J              = IPOINT
C     DO (I = 2,NN)
        I99990       = 2
        I99989       = NN
          DO 6 I       = I99990, I99989
          J            = J + 1
          IF (J .GT. ISTOP) GO TO 12
C     WHILE  (LIST(J) .EQ. IBLANK .AND. J .LE. ISTOP)
    2     IF ( .NOT. (LIST(J) .EQ. IBLANK .AND. J .LE.
     $ ISTOP)) GO TO 3
          J            = J + 1
C     END WHILE
          GO TO 2
    3     CONTINUE
    4     CONTINUE
C     IF (LIST(J) .NE. LOOKUP(I))
          IF ( .NOT. (LIST(J) .NE. LOOKUP(I))) GO TO 5
          J            =  - 1
C     ESCAPE DO
          GO TO 7
C     END IF
    5     CONTINUE
```

```
C     END DO
    6      CONTINUE
    7    CONTINUE
C
C     IF J IS POSITIVE HERE, THE STRING WAS FOUND.
C
C     IF (J .GT. 0)
         IF ( .NOT. (J .GT. 0)) GO TO 8
         CHECK        = .TRUE.
         RETURN
C     END IF
    8    CONTINUE
C     END IF
    9    CONTINUE
C
C     ELSE, LOOK FOR THE NEXT OCCURENCE IN LIST OF THE
C    $         FIRST
C     CHARACTER OF LOOKUP.
C
C     END DO
   10     CONTINUE
   11   CONTINUE
C
C     IF THE OUTER LOOP IS COMPLETED, NO MATCH WAS FOUND.
C
   12   CONTINUE
        CHECK        = .FALSE.
        IPOINT       = ISTART
        RETURN
C
C
  100   FORMAT  ( 100A1 )
C
        END
```

```
      SUBROUTINE  INSERT (NEW, ISTART, ISTOP, LIST, N)
C
C        THIS ROUTINE INSERTS INTO THE DATA STRING 'LIST'
C    $        THE N
C        CHARACTERS PASSED THRU NEW.  START AT POSITION
C    $        ISTART.
C        ISTOP IS INCREASED BY N.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
    $ 50), IPOINT,
    1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
    $ 100), KSNUM
    2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
    $ (2000),
    3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
    $ NKFORM, NOUTS,
    4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
    $ NVALUE, STRING
    5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
    $ (10), IFUNCT
    1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
    $ MLCHARS,
    2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
    $ RETURN,
    3   STAR, X
      DIMENSION   LIST (1), NEW (1), NEWTEMP (100)
      NN          = N
      IF (NN .LE. 0) RETURN
      IF (NUMIN .LE. 0) GO TO 2
        DO 1 J    = 1, NUMIN
        IF  (ISTART .LT. INNUM(1,J)) INNUM (1,J) = INNUM
    $ (1,J) + NN
   1    CONTINUE
   2  DECODE (NN,  100, NEW (1))    (NEWTEMP (I), I=1, NN)
   3    DO 4 I    = 1, NN
        CALL SHIFTI  (NEWTEMP(I), ISTART + I - 1, ISTOP,
    $ LIST (1))
   4    CONTINUE
      ICHARS      = ICHARS + NN
      IF  (IDOLLAR .GT. 0) IDOLLAR = IDOLLAR + NN
      RETURN
C
      ENTRY INSERTN
C
      ENCODE (10,  101, NEWTEMP (100))  NEW (1)
      DECODE (5,  100, NEWTEMP (100))   (NEWTEMP (I), I=1,
    $ 5)
      NN          = 5
      IF (N .GE. 4) GO TO 3
    5 IF (NEWTEMP(2) .NE. IBLANK) GO TO 3
      NN          = NN - 1
        DO 6 I       = 2, NN
        NEWTEMP(I) = NEWTEMP(I + 1)
    6   CONTINUE
      GO TO 5
C
      ENTRY INSERTS
C
      NN            = N
      IF (NN .LE. 0) RETURN
      GO TO 2
C
  100 FORMAT  ( 100A1 )
  101 FORMAT  ( I5 )
C
      END
      FUNCTION ISCANR (LOOK4, ISTART, ISTOP, LIST)
C
C          THIS FUNCTION SCANS THE ARRAY 'LIST' FOR THE
C    3                CHARACTER 'LOOK4'
C          AND RETURNS THE LOCATION IF FOUND.
C          SCAN FROM THE RIGHT.
C
      DIMENSION   LIST (1)
      I           = ISTOP
    1 IF (I .LT. ISTART) GO TO 3
      IF (LIST(I) .EQ. LOOK4) GO TO 3
      I           = I - 1
      GO TO 1
C
      ENTRY ISCANL
C
C          SCAN FROM THE LEFT.
C
      I           = ISTART
    2 IF (I .GT. ISTOP) GO TO 3
      IF (LIST(I) .EQ. LOOK4) GO TO 3
      I           = I + 1
      GO TO 2
    3 ISCANR      = I
      RETURN
      END
      FUNCTION  MATCH (ISTART, ISTOP, LIST)
C
C        THIS FUNCTION FINDS THE CLOSING ).  ISTART IS THE
C    $        KNOWN
C        POSITION OF THE FIRST (.
C
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOLT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION   LIST (1)
      I4          = I3 = ISTART + 1
C
C        I3 IS THE POSITION OF THE NEXT (.
C
    1 I3          = ISCANL(IPUNCT(3), I3, ISTOP, LIST(1))
C
C        I4 IS THE POSITION OF THE NEXT ).
C
      I4          = MATCH = ISCANL(IPUNCT(4), I4, ISTOP,
     $ LIST(1))
C
C        LAST ) IS FOUND WHEN NEXT ( IS TO THE RIGHT OF
C    $        WHEN ISTOP
C        HAS BEEN EXCEEDED.
C
      IF (I3 .GE. I4 .OR. I4 .GT. ISTOP) RETURN
C
C        ACROSS BY PAIRS
C
      I3          = I3 + 1
      I4          = I4 + 1
      GO TO 1
      END
```

```
          FUNCTION  NONI  (LOOK4, ISTART, ISTOP, LIST)
C
C             THIS FUNCTION FINDS THE POSITION OF THE LAST
C      $             (NONR) OR FIRST
C             (NONL) CHARACTER IN THE STRING 'LIST' BETWEEN
C      $             ISTART AND ISTOP
C             WHICH DOES NOT MATCH THE CHARACTER 'LOOK4'.
C
C             SCAN FROM THE RIGHT (LAST).
C
          DIMENSION     LIST (1)
          I             = ISTOP
    1     IF (I .LT. ISTART) GO TO 3
          IF (LIST(I) .NE. LOOK4) GO TO 3
          I             = I - 1
          GO TO 1
C
          ENTRY NONL
C
C             SCAN FROM THE LEFT (FIRST).
C
          I             = ISTART
    2     IF (I .GT. ISTOP) GO TO 3
          IF (LIST(I) .NE. LOOK4) GO TO 3
          I             = I + 1
          GO TO 2
    3     NONR          = I
          RETURN
          END
```

```
      SUBROUTINE  RESETS
C
C        THIS ROUTINE RESETS THE POINTERS AND COUNTERS.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      INTEGER      STRING
        DO 1 I     = 1, 7
        NUMBER(I) = 0
1       CONTINUE
        DO 2 J     = 1, 100
          DO 2 I     = 1, 2
          STRING(I, J) = IBLANK
2         CONTINUE
        DO 3 I     = 1, 4
        NAME(I)    = IBLANK
3       CONTINUE
        DO 4 I     = 1, 1000
        LFOUT(I)   = IBLANK
        LSTATE(I)  = IBLANK
4       CONTINUE
        DO 5 J     = 1, 100
        KFORM(J)   = 0
          DO 5 I     = 1, 3
          KFOUT(I, J) = 0
5         CONTINUE
        DO 6 I     = 1, 4
        ICOUNT(1, I) = 0
6       CONTINUE
      ICHARS     = 0
      IERROR     = 0
      IPROG      = 0
      I9999      = 0
      LCHARS    = 0
      NCARDS    = 0
      NEXT      = 1
      NFORMN    = 0
      NFOUT     = 0
      NKFORM    = 0
      NOUTS     = 0
      NSNUMC    = 0
C        ***
      NSTATN    = 0
      NUMK      = 0
C
      ENTRY RESETX
        DO 7 J    = 1, NUMMAX
          DO 7 I    = 1, 2
          INNUM(I, J) = 0
    7     CONTINUE
      ISNUM     = 0
      ITYPE     = 1
      NVALUE    = 0
      NUMIN     = 0
      RETURN
      END
      SUBROUTINE  SCANREF (N, NQ, NL, NR)
C
C        SCANS BINARY TREE 'REF' FOR VARIABLE N.
C        IF N IS IN REF, ITS SUBSCRIPT IS RETURNED IN NQ.
C        OTHERWISE, THE NEXT LEFT POINTER IS RETURNED IN
C    $           NL
C        OR THE NEXT RIGHT POINTER IS RETURNED IN NR.
C
      COMMON  /SNLIST/  NS, REF (400, 3)
      INTEGER      REF
      I            = 1
    1 IF (N - REF(I, 1)) 2, 3, 4
C
C        CHECK LEFT POINTER
C
    2 IF (REF(I, 2) .EQ. 0) GO TO 5
      I            = REF(I, 2)
      GO TO 1
C
C        N FOUND
C
    3 NQ           = I
      NL           = NR = 0
      RETURN
    4 IF (REF(I, 3) .EQ. 0) GO TO 6
C
C        CHECK RIGHT POINTER
C
      I            = REF(I, 3)
      GO TO 1
    5 NL           = I
C
C        END OF BRANCH.  N IS NOT IN REF.
C
      NR           = 0
      GO TO 7
    6 NR           = I
      NL           = 0
    7 NQ           = 0
      RETURN
      END
      SUBROUTINE  SHIFTR (NEW, ISTART, ISTOP, LIST)
C
C         THIS ROUTINE SHIFTS ALL DATA IN THE LIST FROM
C    $          ISTART THRU
C         ISTOP ONE SPACE TO THE RIGHT.  THE CREATED SPACE
C    $           IS FILLED
C         BY NEW.
C
      DIMENSION   LIST (1)
      I           = ISTOP
    1 LIST(I + 1) = LIST(I)
      I           = I - 1
      IF (I .GE. ISTART) GO TO 1
      LIST(ISTART) = NEW
      ISTOP       = ISTOP + 1
      RETURN
C
      ENTRY SHIFTL
C
C         THIS ROUTINE SHIFTS ALL DATA IN THE LIST FROM
C    $          ISTART THRU
C         ISTOP ONE SPACE TO THE LEFT.  THE CREATED SPACE
C    $          IS FILLED
C         BY NEW.
C         NOTICE... THE VALUE OF ISTOP IS ADJUSTED.
C
      ISTOP       = ISTOP - 1
      IF (ISTART .GT. ISTOP) GO TO 3
        DO 2 I      = ISTART, ISTOP
        LIST(I)     = LIST(I + 1)
    2   CONTINUE
    3 LIST(ISTOP + 1) = NEW
      RETURN
      END
      OVERLAY(CLEAN,1,0)
      PROGRAM  READS
C
C        THIS SUBROUTINE READS THE INPUT FILE AND
C    $          GENERATES THE WORK
C        FILE AND STRINGS FOR LATER PROCESSING.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORM, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      COMMON  /SNLIST/  NS, REF (400, 3)
C
C        IPUNCT   1  2  3  4  5  6  7  8  9 10 11
C                 /  ,  (  )  *  $  .  =  -  +  "
C
      INTEGER      C, END, PROGRAM, REF, STAR, STRING,
     $ TRANSF
      LOGICAL      CHECK, KLIST, KO
      CALL KSET (0)
      ICOUNT(1, 1) = 1
      IF (NCARD .NE. 0) GO TO 3
      ICOUNT(1, 1) = 0
    1 READ (LUIN,  100)  LCARD
      ICOUNT(1, 1) = ICOUNT(1, 1) + 1
      IF   (EOF(LUIN))  82, 2
    2 NCARD      = 1
C
C        CHECK FOR A COMMENT CARD.  IF SO, OUTPUT.
C
    3 IF (IEOF .EQ. 1) GO TO 83
      IF (LCARD(1) .EQ. C .OR. LCARD(1) .EQ. STAR) GO TO 4
C
C        CHECK FOR ALL BLANK CARD.  IF SO, OUTPUT C CARD.
C
```

```
      IF (NONL(IBLANK,1,72,LCARD(1)) .LE. 72) GO TO 6
      ICHARS      = 1
      GO TO 5
    4 ICHARS      = NONB(IBLANK, 2, 72, LCARD(1))
    5 ITYPE       = 0
      LCARD(1)    = 0
      CALL OUTPUT (LCARD(1))
      NCARD       = 0
      GO TO 1
    6 N           = NONL(IBLANK, 1, 5, LCARD(1))
C
C        CHECK FOR STATEMENT NUMBER IN THE FIRST FIVE
C    $        COLUMNS.
C
      IF (N .GT. 5) GO TO 7
C
C        YES.  NOW DETERMINE ITS VALUE.
C
      ISTOP       = 5
      NVALUE      = NUMBS(N, ISTOP, LCARD(1))
      IF (NVALUE .GT. 0) GO TO 7
      PRINT  101, LCARD
      GO TO 4
C
C        TRANSFER THIS RECORD TO LSTATE.
C
    7 ITRANS      = TRANSF(7, 72)
    8 IF (ITRANS .GT. 0) GO TO 11
C
C        READ THE NEXT INPUT RECORD
C
      READ (LUIN,  100)  LCARD
      ICOUNT(1, 1) = ICOUNT(1, 1) + 1
      IF   (EOF(LUIN))  10, 9
    9 NCARD       = NCARD + 1
C
C        IS THIS A CONTINUATION?
C
      IF (LCARD(6) .EQ. INTEGER(1) .OR. LCARD(6) .EQ.
     $ IBLANK .OR.
     1  LCARD(1) .EQ. C .OR. LCARD(1).EQ.STAR) GO TO 11
C
C        YES.  SET UP FOR TRANSFER TO LSTATE.
C
      N           = NONL(IBLANK, 1, 5, LCARD(1))
      IF (N .GT. 5) GO TO 7
      GO TO 11
   10 IEOF        = 1
C
C        THE ENTIRE ARRAY HAS BEEN CONSTRUCTED.  NOW
C    $        IDENTIFY THE
```

```
C        TYPE AND INSERT THE PROPER SPACING.
C
   11 IF (IPROG .NE. 0) GO TO 14
      CALL BLANKS
C
C        STATEMENTS WITH ITYPE = 1, 2, 3, 4, OR 14.
C
      IPOINT      = 1
      J           = IDENT(1)
      IF (J .NE. 45) GO TO 12
C
C        UNIDENTIFIABLE.  THIS IS AN ERROR.
C
      PRINT   102
      IPROG       = 100
      GO TO 14
C
C        IDENTIFIED.
C
   12 ITYPE       = J
      IF (ITYPE .EQ. 14) GO TO 66
      IPROG       = J
      IF (J .NE. 4) GO TO 13
C
C        HERE FOR BLOCK DATA.
C
      CALL INSERT (IBLANK, IPOINT - 4, LCHARS, LSTATE(1),
     $ 1)
      IPOINT      = IPOINT + 1
      GO TO 66
   13 CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 2)
      IPOINT      = IPOINT + 2
      GO TO 65
C
C        START PROCESSING THE ROUTINE STATEMENTS
C
   14 IPOINT      = 1
      J           = IDENT(2)
      IERROR      = J
      ITYPE       = J
      IF (J .LE. 4 .OR. J .GT. 46) GO TO 77
      CALL BLANKS
      IF (J .NE. 46) GO TO 15
C
C        IMPLICIT  (J = 46).
C        INSERT TWO BLANKS.
C
      CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 2)
      IPOINT      = IPOINT + 2
      GO TO 65
   15 CONTINUE
      GO TO  ( 19, 17, 21, 21, 17, 17, 17, 17, 17, 20, 22,
     $ 16, 24, 26,
     1   27, 18, 33, 17, 44, 45, 46, 18, 18, 46, 18, 49,
     $ 50, 46, 46, 17,
     2   17, 17, 56, 17, 17, 57, 17, 17, 61, 76, 63)  J - 4
C
C        MAKE A SPECIAL CHECK FOR DATA STATEMENTS, J = 16.
C        DATA (TEXT(I),I=1,9) / LIST / IS OK.  MUST CHECK
C    $          FOR THE
C        RELATIVE POSITIONS OF THE MATCHING ( ), I4, AND
C    $          THE =, I8.
C
   16 IF (LSTATE(IPOINT) .NE. IPUNCT(3)) GO TO 18
      I4           = MATCH(IPOINT, ICHARS, LSTATE(1))
      I8           = ISCANL(IPUNCT(8), IPOINT + 1, ICHARS,
     $ LSTATE(1))
      IF (I8 .LT. I4) GO TO 23
      GO TO 62
C
C        CHECK FOR ( OR = FOLLOWING THE TYPE WORD JUST
C    $          IDENTIFIED.
C
   17 IF (LSTATE(IPOINT) .EQ. IPUNCT(3)) GO TO 62
   18 IF (LSTATE(IPOINT) .EQ. IPUNCT(8)) GO TO 62
C
C        NOW WORK THE STATEMENTS.
C
      GO TO  ( 19, 61, 21, 21, 21, 21, 21, 21, 21, 20, 22,
     $ 23, 24, 26,
     1   27, 31, 33, 43, 44, 45, 46, 47, 47, 46, 47, 49,
     $ 50, 46, 46, 52,
     2   52, 53, 56, 57, 57, 57, 58, 57, 61, 76, 63)  J - 4
   19 CALL INSERT (IBLANK, IPOINT - 1, ICHARS, LSTATE(1),
     $ 2)
      IPOINT       = 1 + ISCANL(IPUNCT(1), IPOINT + 2,
     $ ICHARS, LSTATE(1))
      GO TO 61
C
C        SET PRECISION TO DOUBLE.
C
   20 J            = 10
C
C        STORE THE TYPE STATEMENTS IN THE ARRAY STRING.
C
   21 CALL STORE (J - 6)
      GO TO 63
C
C        EQUIVALENCE (J = 15).
```

```
C           INSERT A BLANK.
C
      22  CALL INSERT (IBLANK, IPOINT - 1, LCHARS, LSTATE(1),
        $ 1)
          GO TO 65
C
C           DATA STATEMENTS (J = 16).
C
      23  CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 4)
          IPOINT      = IF(IN) + 4
          GO TO 65
C
C           FORMAT (J = 17).
C
      24  ICHARS      = ICHARS - IPOINT
          IF ( .NOT. KO(NVALUE)) GO TO 69
          IN          = KFOUT(2, NFOUT)
            DO 25 II  = IN, 1000, 10
            I2        = MIN0(IPOINT + 99, LCHARS - 1)
            IC        = I2 + 1 - IPOINT
            IF (IC .LE. 0) GO TO 69
            ENCODE (IC,  100, LFOUT (II))    (LSTATE (I),
        $ I=IPOINT, I2)
            IPOINT    = IPOINT + 100
      25    CONTINUE
          GO TO 69
C
C           DO STATEMENT (DO 1 I = 1,11)  (J = 13).
C
      26  N           = NUMBS(IPOINT, LCHARS, LSTATE(1))
          IF (N .LE. 0) GO TO 62
          IF ( .NOT. KLIST(IPOINT,N)) GO TO 63
          CALL KU (N)
          CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
          GO TO 64
C
C           COMPUTED GO TO (J = 19).
C
      27  CALL INSERT (IBLANK, IPOINT - 1, LCHARS, LSTATE(1),
        $ 2)
          CALL INSERT (IBLANK, IPOINT - 3, LCHARS, LSTATE(1),
        $ 1)
          IPOINT      = IPOINT + 3
          N           = NUMBS(IPOINT, LCHARS, LSTATE(1))
C
C           THERE MUST BE A STATEMENT NUMBER IN THE FIRST
C       $          POSITION.
C
          IF (N .LE. 0) GO TO 62
          GO TO 29
```

```
28   N            = NUMBS(IPOINT, LCHARS, LSTATE(1))
     IF (N .LE. 0) GO TO 30
29   IF ( .NOT. KLIST(IPOINT,N)) GO TO 61
     CALL KU (N)
     IPOINT       = IPOINT + 1
     IF (LSTATE(IPOINT-1) .EQ. IPUNCT(4)) GO TO 30
     GO TO 28
30   IF   (LSTATE(IPOINT) .EQ. IPUNCT(2)) IPOINT = IPOINT
    $ + 1
     GO TO 61
C
C        GO TO  (J = 20).
C
31   CALL INSERT (IBLANK, IPOINT - 2, LCHARS, LSTATE(1),
    $ 1)
     IPOINT       = IPOINT + 1
     N            = NUMBS(IPOINT, LCHARS, LSTATE(1))
     IF (N .GT. 0) GO TO 32
C
C        ASSIGNED GO TO (SUBSECTION).
C
     CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 2)
     IPOINT       = IPOINT + 2
     IPOINT       = ISCANL(IPUNCT(3), IPOINT, LCHARS,
    $ LSTATE(1))
     CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
     IPOINT       = IPOINT + 2
     GO TO 28
32   IF ( .NOT. KLIST(IPOINT,N)) GO TO 65
     CALL KU (N)
     ISTOP        = ICHARS = INNUM(1, 1) - 1
     GO TO 66
C
C        IF STATEMENT (J = 21).
C
33   CALL INSERT (IBLANK, IPOINT - 1, LCHARS, LSTATE(1),
    $ 1)
     IPOINT       = 1 + MATCH(IPOINT, LCHARS, LSTATE(1))
     CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
     IPOINT       = IPOINT + 1
     N            = NUMBS(IPOINT, LCHARS, LSTATE(1))
C
C        ARITHMETIC OR LOGICAL?
C
     IF (N .LE. 0) GO TO 36
     GO TO 35
34   IPOINT       = IPOINT + 1
     N            = NUMBS(IPOINT, LCHARS, LSTATE(1))
     IF (N .LE. 0) GO TO 64
C
```

```
C           ARITHMETIC.  STORE THE STATEMENT NUMBERS.
C
      35  IF ( .NOT. KLIST(IPOINT,N)) GO TO 65
          CALL KU (N)
          GO TO 34
C
C           LOGICAL.  IDENTIFY THE CONDITIONAL STATEMENT.
C
      36  JJ         = IDENT(2)
          IF (JJ .LE. 18 .OR. JJ .GT. 45) GO TO 77
          GO TO   ( 27, 39, 33, 38, 44, 67, 46, 39, 39, 46, 38,
        $ 38, 50, 46,
        1   46, 38, 38, 38, 37, 38, 38, 57, 38, 38, 61, 76,
        $ 63)  JJ - 18
C
C           CHECK FOR AN ASTERISK.
C
      37  IF (LSTATE(IPOINT) .EQ. IFUNCT(5)) GO TO 74
C
C           CHECK FOR A ( OR AN = FOLLOWING THE IDENTIFIER
C       $        NAME.
C
      38  IF (LSTATE(IPOINT) .EQ. IFUNCT(3)) GO TO 63
      39  IF (LSTATE(IPOINT) .EQ. IFUNCT(8)) GO TO 63
C
          GO TO   ( 27, 31, 33, 40, 44, 67, 46, 47, 47, 46, 47,
        $ 49, 50, 46,
        1   46, 52, 52, 55, 56, 57, 57, 57, 58, 57, 61, 76,
        $ 63)  JJ - 18
C
C           CALL SUBROUTINE  (J = 22).
C
      40  CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
          IPOINT     = IPOINT + 2
C
C           LOOK FOR NONSTANDARD RETURNS.
C
          IP1        = IPOINT + 1
          IP2        = ICHARS
          IF ( .NOT. CHECK("RETURNS",7,IP1,IP2,LSTATE(1),IP3))
        $ GO TO 65
          IPOINT     = 1 + ISCANR(IFUNCT(3), IPOINT, LCHARS,
        $ LSTATE(1))
          N          = NUMBS(IPOINT, LCHARS, LSTATE(1))
          IF (N .LE. 0) GO TO 43
          GO TO 42
      41  IPOINT     = IPOINT + 1
          N          = NUMBS(IPOINT, LCHARS, LSTATE(1))
          IF (N .LE. 0) GO TO 43
      42  IF ( .NOT. KLIST(IPOINT,N)) GO TO 43
      CALL KU (N)
      GO TO 41
   43 IPOINT     = IP1 - 1
      GO TO 65
C
C        ASSIGN STATEMENT (J = 23).
C
   44 N          = NUMBS(IPOINT, LCHARS, LSTATE(1))
      IF (N .LE. 0) GO TO 62
      IF ( .NOT. KLIST(IPOINT,N)) GO TO 65
      CALL KU (N)
      CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
      IPOINT     = IPOINT + 1
      IF ( .NOT. CHECK(2HTO,2,IPOINT,LCHARS,LSTATE(1),
     $ IPOINT)) GO TO 65
      CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 2)
      GO TO 66
C
C        CONTINUE  (J = 24).
C        OMIT IF NO STATEMENT LABEL (NVALUE = 0).
C
   45 IF (NVALUE .LE. 0) GO TO 69
      GO TO 67
C
C        READ (XX,YY) LIST  (J = 25).
C        WRITE (XX,YY) LIST  (J = 28).
C        DECODE (XX,YY,V) LIST  (J = 32).
C        ENCODE (XX,YY,V) LIST  (J = 33).
C
   46 CALL INSERT (IBLANK, IPOINT - 1, LCHARS, LSTATE(1),
     $ 1)
      IPOINT     = IPOINT + 1
      I          = 1 + MATCH(IPOINT, LCHARS, LSTATE(1))
      CALL INSERT (IBLANK, I, LCHARS, LSTATE(1), 2)
      IPOINT     = ISCANL(IPUNCT(2), IPOINT, LCHARS,
     $ LSTATE(1)) + 1
C
C        READ XX, LIST  (J = 26).
C        PRINT XX, LIST  (J = 27).
C        PUNCH XX, LIST  (J = 29).
C
C        CHECK FOR LIST - DIRECTED IO (READ *, PRINT *,
C    $         ETC).
C
   47 IF (LSTATE(IPOINT) .EQ. IPUNCT(5)) GO TO 65
      N          = NUMBL(IPOINT, LCHARS, LSTATE(1))
      IF (N .GT. 0) GO TO 48
C
C        NO FORMAT LABEL. ASSUME THIS TO BE A NAMELIST
C    $         READ,
C          WRITE, OR PUNCH STATEMENT.  INSERT A BLANK BEFORE
C    $          THE
C          NAMELIST NAME.
C
       CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
       IPOINT      = IPOINT + 1
       GO TO 65
   48  IF ( .NOT. KLIST(IPOINT,N)) GO TO 65
       CALL KF (N)
       CALL KU (N)
       GO TO 65
C
C          BUFFER IN (XX,YY,V) LIST  (J = 30).
C
   49  CALL INSERT (IBLANK, IPOINT - 3, LCHARS, LSTATE(1),
     $ 1)
       GO TO 51
C
C          BUFFER OUT (XX,YY,V) LIST  (J = 31).
C
   50  CALL INSERT (IBLANK, IPOINT - 4, LCHARS, LSTATE(1),
     $ 1)
   51  CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
       IPOINT      = IPOINT + 1
       IPOINT      = 1 + MATCH(IPOINT, LCHARS, LSTATE(1))
       GO TO 61
C
C          STOP STATEMENT  (J = 34).
C          ENTRY STATEMENT  (J = 35).
C
   52  CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
       GO TO 66
C
C          RETURN  (J = 36).
C
   53  IF (ICHARS .GT. 6) GO TO 60
C
C          CHECK FOR A MULTI - STATEMENT RECORD.
C
       IF (IDOLLAR .GT. 0) GO TO 74
       IPOINT      = 1
   54  LCHARS      = IPOINT - 1
C
C          SEE IF THE NEXT RECORD IS AN END STATEMENT.
C
       IP1         = 7
       IP2         = 72
       IF ( .NOT. CHECK(END,3,IP1,IP2,LCARD(1),IP3)) GO TO
     $ 66
       IF (NONL(IBLANK,IP3,IP2,LCARD(1)) .GT. IP2) GO TO 73
```

```
      GO TO 66
C
C        IF STATEMENTS
C
   55 IF (ICHARS .GT. IPOINT) GO TO 61
      IF (IDOLLAR .GT. 0) GO TO 61
      IPOINT     = IPOINT - 6
      GO TO 54
C
C        USE (LFN)  (J = 37).
C
   56 IPOINT     = 1
      GO TO 65
C
C        END FILE  (J = 38).
C        REWIND  (J = 39).
C        BACKSPACE  (J = 40).
C        PAUSE  (J = 42).
C
   57 CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 1)
      GO TO 66
C
C        J = 41.  SUPPRESS THE WORD 'TYPE'.
C
   58   DO 59 I    = 1, 4
        CALL SHIFTL (IBLANK, 1, LCHARS, LSTATE(1))
   59   CONTINUE
      GO TO 14
C
C        CHANGE A NUMBERED RETURN TO J = 44.
C
   60 J          = 44
      IERROR     = J
      ITYPE      = J
C
C        NAMELIST  (J = 43).
C
   61 CALL INSERT (IBLANK, IPOINT, LCHARS, LSTATE(1), 2)
      IPOINT     = IPOINT + 2
      GO TO 65
C
C        REPLACEMENT (X = V)  (J = 45).
C
   62 J          = 45
      ITYPE      = J
C
C        PROCESS EQUAL SIGN.
C
   63 CALL ALIGN
   64 IPOINT     = 2
```

```
C
   65  CALL SPACOUT
C
   66  IF (NVALUE .LE. 0) GO TO 68
   67  NSTATN      = NSTATN + 1
       IF (NSTATN .GT. MNSTATE) GO TO 78
C
C         NVALUE IS THE ORIGINAL STATEMENT NUMBER.
C         NSNUMC IS THE NEW STATEMENT NUMBER.
C
       KSNUM(1, NSTATN) = NVALUE
       NSNUMC      = NSNUMC + 1
       ISNUM       = NVALUE
       KSNUM(2, NSTATN) = NSNUMC
   68  CALL OUTPUT (LSTATE(1))
   69  CALL RESETX
       NCARD       = 1
       IF (IDOLLAR .LE. 0) GO TO 71
C
C         MULTIPLE STATEMENTS SEPARATED BY A DOLLAR SIGN.
C         SHIFT LEFT AND GO AGAIN.
C
       LCHARS      = LCHARS - IDOLLAR
       ICHARS      = LCHARS
         DO 70 I   = 1, LCHARS
         LSTATE(I) = LSTATE(I + IDOLLAR)
         LSTATE(I + IDOLLAR) = 0
   70    CONTINUE
       GO TO 14
C
C         CLEAN THE ARRAY AND RETURN TO START THE NEXT
C    $         RECORD.
C
   71    DO 72 I   = 1, LCHARS
         LSTATE(I) = IBLANK
   72    CONTINUE
       LCHARS      = 0
       IF (ITRANS .EQ. 0) GO TO 3
       ITRANS      = TRANSF(ITRANS, 72)
       NCARD       = 1
       GO TO 8
C
C         END PROCESSING FOLLOWING A RETURN STATEMENT.
C
   73  NCARD       = 0
   74  IF (NVALUE .LE. 0) GO TO 75
       NSTATN      = NSTATN + 1
       IF (NSTATN .GT. MNSTATE) GO TO 76
       KSNUM(1, NSTATN) = NVALUE
       NSNUMC      = NSNUMC + 1
```

```
      ISNUM      = NVALUE
      KSNUM(2, NSTAIN) = NSNUMC
   75 CALL OUTPUT (LSTATE(1))
      GO TO 83
C
   76 IF (LCHARS .GT. 3) GO TO 62
      ICOUNT(1, 1) = ICOUNT(1, 1) - 1
      GO TO 83
C
   77 PRINT  103, IERROR, (LSTATE(I), I=1, LCHARS)
      IF (ITYPE .NE. 45) GO TO 62
      GO TO 71
   78 PRINT  104, MNSTATE
      PRINT  105, (LSTATE(I), I=1, LCHARS)
C
C        DUMP THE REMAINDER OF THIS ROUTINE.
C
   79 PRINT  106, LCARD, NAME
      REWIND LUSTATE
      IP2        = 72
      N          = 7
C
C        CHECK FOR AN END STATEMENT.
C
   80 IF (CHECK(END,3,N,IP2,LCARD(1),IP3)) GO TO 81
C
C        CHECK FOR A DOLLAR SIGN INDICATING A MULTIPLE
C    *         STATEMENT.
C
      N          = ISCANL(IPUNCT(6), N + 1, 72, LCARD(1)) +
     $ 1
      IF (N .LE. 72) GO TO 80
C
C        READ THE NEXT RECORD.
C
      READ (LUIN,  100)  LCARD
      ICOUNT(1, 1) = ICOUNT(1, 1) + 1
      IF  (EOF(LUIN))  82, 79
C
C        END FOUND.  RESET AND START THE NEXT ROUTINE.
C
   81 CALL RESETS
      GO TO 1
C
C        EOF.  TERMINATE.
C
   82 IEOF       = 1
      ICOUNT(1, 1) = ICOUNT(1, 1) - 1
   83 CONTINUE
C
```

```
 100  FORMAT  ( 100A1 )
 101  FORMAT  (  *0ERROR IN THE FIRST FIVE COLUMNS OF *,
    $ 80A1 /
    1    * THIS RECORD HAS BEEN LEFT IN THE FINAL ROUTINE
    $ AS A COMMENT.*
    2    )
 102  FORMAT  (  *0NO PROGRAM, SUBROUTINE, FUNCTION, OR
    $ BLOCK DATA STA*
    1    *TEMENT FOUND FOR THIS ROUTINE.* /  * CHECK THE
    $ FIRST AND LAST*
    2    * TWO RECORDS BEFORE COMPILING.*  )
 103  FORMAT  (  *0ERROR IN THE FOLLOWING STATEMENT.
    $ ITYPE = *, I5 /
    1    (20X, 100A1) )
 104  FORMAT  (  *0THE ARRAY KSNUM IS FULL.  THE NUMBER OF
    $ EXECUTABLE *
    1    *STATEMENT NUMBERS EXCEEDED *, I5 )
 105  FORMAT  (  *0THE PREVIOUS ERROR FORCED THE
    $ TERMINATION OF PROCES*
    1    *SING OF THE INPUT FOR THIS ROUTINE ON STATEMENT*
    $ / (20X,
    2    100A1) )
 106  FORMAT  (  *0THIS INPUT RECORD NOT PROCESSED  *,
    $ 80A1,
    1    *   FOR ROUTINE *, 4A1 )
C
      END
      SUBROUTINE  BLANKS
C
C         THIS ROUTINE SUPPRESSES ALL BLANKS IN THE ARRAY
C    $         LIST EXCEPT
C         THOSE IN HOLLERITH TYPE STATEMENTS.
C         IT ALSO FINDS THE $ SEPARATORS IN MULTIPLE
C    $         STATEMENT RECORDS.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4  NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5  (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3  STAR, X
      DIMENSION    LIST (1)
      INTEGER      C, DLQ (3), H, HLR (3), STAR, X
      LOGICAL      DOLLAR, VALID
      EQUIVALENCE  (LIST(1), LSTATE(1))
      DATA     DLQ / 1H", 1H*, 1H' /
      DATA     HLR / 1HH, 1HL, 1HR /
C
C         IPUNCT  1  2  3  4  5  6  7  8  9 10 11
C                 /  ,  (  )  *  $  .  =  -  +  "
C
C         LOOK FOR HOLLERITH LITERALS IN DATA, FORMAT, IF,
C    $         CALL,
C         PRINT *, WRITE (LU,*), AND REPLACEMENT
C    $         STATEMENTS.
C         FIX THESE, THEN REMOVE ALL BLANKS.
C
      INPOINT   = IPOINT
      IDOLLAR   = 0
      ISTOP     = NCHR(IBLANK, 1, LCHARS, LIST)
      LCHARS    = ICHARS = ISTOP
      VALID     = .FALSE.
C
C         GENERATE STARTING LOCATION.
```

```
C
C         CASE OF (ITYPE)
C
C         CASE (5,6,7,8,9,10,11,12,13,15)
C
       IF (ITYPE .NE. (5) .AND. ITYPE .NE. (6) .AND. ITYPE
     $ .NE. (7)
     1   .AND. ITYPE .NE. (8) .AND. ITYPE .NE. (9) .AND.
     $ ITYPE .NE. (10)
     2   .AND. ITYPE .NE. (11) .AND. ITYPE .NE. (12) .AND.
     $ ITYPE.NE.(13)
     3   .AND.ITYPE.NE.(15)) GO TO 1
C
C         SPECIFICATION STATEMENTS
C
       IPOINT     = 1
       ISTART     = IPOINT
       GO TO 8
C
C         CASE (16)
C
    1  CONTINUE
       IF (ITYPE .NE. (16)) GO TO 2
C
C         DATA
C
       IPOINT     = ISCANL(IPUNCT(1), 1, ISTOP, LIST)
       ISTART     = IPOINT
       GO TO 8
C
C         CASE (17,21)
C
    2  CONTINUE
       IF (ITYPE .NE. (17) .AND. ITYPE .NE. (21)) GO TO 3
C
C         FORMAT, IF
C
       IPOINT     = ISCANL(IPUNCT(3), 1, ISTOP, LIST)
       ISTART     = IPOINT
       GO TO 8
C
C         CASE (22)
C
    3  CONTINUE
       IF (ITYPE .NE. (22)) GO TO 4
C
C         CALL
C
       IPOINT     = 2 + ISCANL(HLP(2), 1, ISTOP, LIST)
       ISTART     = IPOINT
```

```
      GO TO 8
C
C        CASE (27)
C
    4 CONTINUE
      IF (ITYPE .NE. (27)) GO TO 5
C
C        PRINT *
C
      IPOINT     = ISCANL(IPUNCT(2), 1, ISTOP, LIST)
      ISTART     = IPOINT
      GO TO 8
C
C        CASE (28)
C
    5 CONTINUE
      IF (ITYPE .NE. (28)) GO TO 6
C
C        WRITE (LU,*)
C
      IPOINT     = ISCANL(IPUNCT(4), 1, ISTOP, LIST)
      ISTART     = IPOINT
      GO TO 8
C
C        CASE (45)
C
    6 CONTINUE
      IF (ITYPE .NE. (45)) GO TO 7
C
C        REPLACEMENT
C
      IPOINT     = ISCANL(IPUNCT(8), 1, ISTOP, LIST)
      ISTART     = IPOINT
      GO TO 8
C
C        CASE ELSE
C
    7 CONTINUE
      IPOINT     = ISTOP
      ISTART     = 1
C
C        END CASE
C
    8 CONTINUE
C
C        NOW SCAN FOR A ;, WHICH MIGHT INDICATE A MULTI -
C    *         STATEMENT
C        RECORD.
C
C        IF A ; IS LESS THAN FOUR CHARACTERS FROM THE END
C    *         OF THE RECORD
```

```
C          IT CANNOT BE A SEPARATOR.
C
        IQUIT        = ISTOP - 4
          DO 9 I     = ISTART, IQUIT
          IF (LIST(I) .NE. IFUNCT(6)) GO TO 9
          IF ( .NOT. (DOLLAR(LIST,I,IPOINT,ISTOP,ITYPE,
     $ LCHARS))) GO TO 9
          IDOLLAR    = I
          ICHARS     = ISTOP = I - 1
          GO TO 10
    9     CONTINUE
C
   10   CONTINUE
C
C          ROLE OF $ DETERMINED.
C          NOW SCAN FOR SPECIAL CHARACTERS
C          SCAN ONLY DATA, FORMAT, AND EXECUTABLE
C    $           STATEMENTS.
C
        IF (ITYPE .NE. 16 .AND. ITYPE .NE. 17 .AND. ITYPE
     $ .NE. 21 .AND.
     1   ITYPE .NE. 22 .AND. ITYPE .NE. 27 .AND. ITYPE .NE.
     $ 28 .AND.
     2   ITYPE.NE.29.AND.ITYPE.NE.45) GO TO 23
C
C     ****************************************************
C    $          ***************
C
C      LOOK FOR H, L, OR R
C
        ASSIGN 13    TOIVLD
        ASSIGN 14    TOIDLM
        LDL          = 1
        IMIN         = ISTART
C
C      UNTIL  (LDL .GT. 3)
C
   11   IF (LDL .GT. 3) GO TO 16
        IPOINT       = ISTART
C
C      WHILE (IPOINT .LT. ISTOP)
C
   12   IF (IPOINT .GE. ISTOP) GO TO 15
        IRIGHT       = ISTOP
        ILEFT        = ISCANL(HLR(LDL), IPOINT + 1, ISTOP,
     $ LIST)
        IPOINT       = ILEFT + 1
        IF (ILEFT .GE. ISTOP) GO TO 15
        CALL QDIGIT (JV, ILEFT, IMIN, LIST, N)
```

```
      IF (N .LE. 0) GO TO 12
C
C     INVOKE VALIDATE (IV,IMIN,ITYPE,VALID)
C
      GO TO 28
 13   CONTINUE
      IF ( .NOT. (VALID)) GO TO 12
      IRIGHT     = ILEFT + N + 1
C
C     INVOKE PROTECT (ILEFT,IRIGHT,IPOINT)
C
      GO TO 26
 14   CONTINUE
C
C        END WHILE
C
      GO TO 12
 15   CONTINUE
      LDL        = LDL + 1
C
C     END UNTIL
C
      GO TO 11
 16   CONTINUE
C
C     ****************************************************
C   *          **************
C
C          LOOK FOR ", *, OR '
C
      ASSIGN 20   TO IDELK
      ASSIGN 19   TO IVLD
      LDL         = 1
C
C          UNTIL (LDL .GT. 3)
C
 17   IF (LDL .GT. 3) GO TO 22
      IPOINT     = ISTART
C
C          WHILE (IPOINT .LT. ISTOP)
C
 18   IF (IPOINT .GE. ISTOP) GO TO 21
      IRIGHT     = ISTOP + 1
      ILEFT      = ISCANL(DLQ(LDL), IPOINT + 1, ISTOP,
    $ LIST)
      IMIN       = IPOINT + 1
      IPOINT     = ILEFT + 1
      IF (ILEFT .GE. ISTOP) GO TO 21
      IRIGHT     = ISCANL(DLQ(LDL), ILEFT + 1, ISTOP, LIST)
      IF (IRIGHT .GT. ISTOP) GO TO 18
      IV         = ILEFT - 1
C
C        INVOKE VALIDATE (IV,IMIN,ITYPE,VALID)
C
      GO TO 28
   19 CONTINUE
      IF ( .NOT. (VALID)) GO TO 18
C
C        INVOKE PROTECT (ILEFT,IRIGHT,IPOINT)
C
      GO TO 26
   20 CONTINUE
      IF (LDL .NE. 3) GO TO 18
C
C        CHANGE ' TO " AND PROCEED.
C
      LIST(ILEFT) = LIST(IRIGHT) = IFUNCT(11)
C
C        END WHILE
C
      GO TO 18
   21 CONTINUE
      LDL        = LDL + 1
C
C        END UNTIL
C
      GO TO 17
   22 CONTINUE
C
C        END IF
C
   23 CONTINUE
C
C        AFTER THE HOLLERITH LITERALS HAVE BEEN ALTERED,
C        SQUEEZE OUT ANY REMAINING BLANKS.
C
      I          = 1
C
C        UNTIL (SPRESS (I,ISTOP,LIST) .NE. 0.0)
C
   24 IF (SPRESS(I,ISTOP,LIST) .NE. 0.0) GO TO 25
      I          = I + 1
C
C        END UNTIL
C
      GO TO 24
   25 CONTINUE
C
C        DONE
C
```

```
      ICHARS      = ISTOP
      IPOINT      = INPOINT
      IF  (IDOLLAR .LE. 0) LCHARS = ISTOP
      RETURN
C
C        INTERNAL SUBROUTINES
C
C    *****************************************************
C    $          **************
C
C        PROCEDURE PROTECT (ILEFT,IRIGHT,IPOINT)
C
C        ALTER THE CHARACTERS BETWEEN ILEFT AND IRIGHT.
C
   26 CONTINUE
C
C        DO (I = ILEFT + 1,IRIGHT - 1)
C
      I99920      = ILEFT + 1
      I99919      = IRIGHT - 1
       DO 27 I    = I99920, I99919
        LIST(I)   = LIST(I) + 1
   27   CONTINUE
      IPOINT      = IRIGHT
      GO TO  IDLM, ( 20, 14)
C        END PROTECT
C
C    *****************************************************
C    $          **************
C
C        PROCEDURE VALIDATE (IV,IMIN,ITYPE,VALID)
C
C        IF AN APPARENTLY VALID CONSTRUCT IS FOUND, .
C    $         ATTEMPT TO VERIFY
C        BY CHECKING THE PRECEDING CHARACTERS, BEGINNING
C    $         WITH
C        POSITION IV.
C
   28 CONTINUE
      I           = NONE(IBLANK, IMIN, IV, LIST)
C
C        CASE OF (ITYPE)
C
C        CASE (16)
C
      IF (ITYPE .NE. (16)) GO TO 29
C
C        DATA
C
      VALID       = LIST(I).EQ.IPUNCT(1).OR.LIST(I)
    * .EQ.IPUNCT(2).OR.LIST(
     1   I).EQ.IFUNCT(5)
         GO TO 35
C
C           CASE (17)
C
   29    CONTINUE
         IF (ITYPE .NE. (17)) GO TO 30
C
C           FORMAT
C
         VALID     = LIST(I).EQ.IFUNCT(1).OR.LIST(I)
     $ .EQ.IFUNCT(2).OR.LIST(
     1   I).EQ.IFUNCT(3).OR.LIST(I).EQ.X.OR.LIST(I)
     $ .EQ.IFUNCT(5).OR.LIST
     2   (I).EQ.IFUNCT(11)
         GO TO 35
C
C           CASE (21,22)
C
   30    CONTINUE
         IF (ITYPE .NE. (21) .AND. ITYPE .NE. (22)) GO TO 31
C
C           IF & CALL
C
         VALID       = LIST(I).EQ.IFUNCT(2).OR.LIST(I)
     $ .EQ.IFUNCT(3).OR.LIST(
     1   I).EQ.IPUNCT(7)
         GO TO 35
C
C           CASE (27)
C
   31    CONTINUE
         IF (ITYPE .NE. (27)) GO TO 32
C
C           PRINT *
C
         VALID       = LIST(I).EQ.IFUNCT(2)
         GO TO 35
C
C           CASE (28)
C
   32    CONTINUE
         IF (ITYPE .NE. (28)) GO TO 33
C
C           WRITE (LU,*)
C
         VALID       = LIST(I).EQ.IFUNCT(2).OR.LIST(I)
     $ .EQ.IPUNCT(4)
         GO TO 35
```

```
C
C          CASE (45)
C
  33  CONTINUE
      IF (ITYPE .NE. (45)) GO TO 34
C
C          REPLACEMENT
C
      VALID      = LIST(I).EQ.TRUNCT(8)
      GO TO 35
C
C          CASE ELSE
C
  34  CONTINUE
      VALID      = .FALSE.
C          END CASE
  35  CONTINUE
      GO TO  IVLC, ( 19, 13)
C          END VALIDATE
C
      END
      LOGICALFUNCTION  DOLLAR (LIST, I, IPOINT, ISTOP,
     $ ITYPE, LCHARS)
C
C        IF THE $ IS BETWEEN PARENTHESES OR THE STRING OF
C    $         CHARACTERS
C        FOLLOWING IS NOT A VALID STATEMENT, IT CANNOT BE
C    $         A SEPARATOR.
C        DOLLAR DECIDES WHETHER THE $ AT POSITION I IS A
C    $         SEPARATOR
C        OR PART OF A HOLLERITH STRING.
C
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOLT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
C
C        IPUNCT   1  2  3  4  5  6  7  8  9 10 11
C                 /  ,  (  )  *  $  .  =  -  +  "
C
      DIMENSION    LIST (1)
      DOLLAR     = .FALSE.
C
C        CASE OF (ITYPE)
C
C        CASE (16)
C        DATA
C
      IF (ITYPE .NE. 16) GO TO 1
C        IF THE PRECEDING NONBLANK CHARACTER IS NOT A /,
C    $         THIS $
C        IS NOT A SEPARATOR.
C
      ILEFT      = NONR(IBLANK, 1, I, LIST)
      IF (LIST(ILEFT) .NE. IPUNCT(1)) GO TO 9
C
C        OTHERWISE, THE $ IS A SEPARATOR.
C
      GO TO 8
    1 CONTINUE
C
C        CASE (17,21)
C        FORMAT, IF
C
      IF (ITYPE .NE. (17) .AND. ITYPE .NE. (21)) GO TO 2
      IRIGHT     = MATCH(IPOINT, ISTOP, LIST)
      IF (I .LT. IRIGHT) GO TO 9
```

```
C
      GO TO 6
C
C        CASE (22)
C        CALL
C
    2 CONTINUE
      IF (ITYPE .NE. (22)) GO TO 3
      ILEFT      = ISCANL(IFUNCT(3), IPOINT, ISTOP, LIST)
C
C        IF THE $ IS BETWEEN MATCHING PARENTHESES, IT
C    *         CANNOT BE A
C        SEPARATOR.  OTHERWISE, IT MUST BE.
C
      IF (ILEFT .GE. I) GO TO 7
      IRIGHT     = MATCH(ILEFT, ISTOP, LIST)
      IF (I .GT. ILEFT .AND. I .LT. IRIGHT) GO TO 9
C
      GO TO 7
C
C        CASE (27,45)
C        PRINT *, REPLACEMENT
C
    3 CONTINUE
      IF (ITYPE .NE. (27) .AND. ITYPE .NE. (45)) GO TO 5
C
C        IF THE $ IS BETWEEN QUOTES, IT IS NOT A
C    *         SEPARATOR.
C
      ILEFT      = ISCANL(IFUNCT(11), IPOINT + 1, I, LIST)
      IF (ILEFT .GE. I) GO TO 4
      IRIGHT     = ISCANL(IFUNCT(11), ILEFT + 1, ISTOP,
     * LIST)
      IF (I .GT. ILEFT .AND. I .LT. IRIGHT) GO TO 9
C
    4 CONTINUE
      IF (ITYPE .NE. 45) GO TO 6
      KQ         = ISCANL(IFUNCT(8), I + 1, ISTOP, LIST)
      IF (KQ .LT. ISTOP .AND. KQ .GE. I+2) GO TO 7
C
C        IF ANOTHER = FOUND, THE $ IS PROBABLY
C        A SEPARATOR
C
C
C        CASE (5,6,7,8,9,10,11,12,13,15)
C
    5 CONTINUE
      IF (ITYPE .NE. (5) .AND. ITYPE .NE. (6) .AND. ITYPE
     $ .NE. (7)
     1   .AND. ITYPE .NE. (8) .AND. ITYPE .NE. (9) .AND.
     $ ITYPE .NE. (10)
     2     .AND. ITYPE .NE. (11) .AND. ITYPE .NE. (12) .AND.
     $ ITYPE.NE.(13)
     3     .AND.ITYPE.NE.(15)) GO TO 6
C
C          SPECIFICATION STATEMENTS.  $ CAN ONLY BE A
C    $          SEPARATOR.
C
      GO TO 7
C
C          END CASE
C
    6 CONTINUE
C
C          TRY TO IDENTIFY THE STRING FOLLOWING THE $
C
      JCHARS     = ICHARS
      JPOINT     = IPOINT
      IPOINT     = NCNL(IBLANK, I + 1, LIST)
      ICHARS     = MIN0(IPOINT + 20, LCHARS)
      JTYPE      = IDENT(2)
      IPOINT     = JPOINT
      ICHARS     = JCHARS
      IF (JTYPE .LT. 17 .OR. JTYPE .GT. 45) GO TO 9
C
      IF (JTYPE .NE. 45) GO TO 7
      IQ         = ISCANL(IPUNCT(8), I + 1, ISTOP, LIST)
      IF (IQ .GE. ISTOP) GO TO 9
C
C          THE STRING FOLLOWING THE $ IS APPARENTLY A VALID
C          STATEMENT.  THEREFORE, ASSUME THE $ TO BE A
C    $          SEPARATOR.
C
    7 CONTINUE
    8 DOLLAR     = .TRUE.
    9 RETURN
      END
      FUNCTION  IDENT (N)
C
C        THIS ROUTINE MATCHES CHARACTER STRINGS IN THE
C    $         LIST ISTATE TO
C        A MASTER LIST, IA, WHERE:
C              IA(1,X)  IF THE CHARACTER IN THE LIST LSTATE
C    $         EXCEEDS THE
C                       MATCH CHARACTER IN IA(2,X), THEN
C    $         JUMP TO THIS
C                       POSITION.  ELSE, EXIT WITH IDENT =
C    $         45.
C              IA(2,X)  IS THE MATCH CHARACTER.
C              IA(3,X)  IS THE END CODE WHEN A MATCH OCCURS.
C                       IF NEGATIVE, THIS MAY BE THE END OF
C    $         THE STRING,
C                       BUT IT COULD CONTINUE TO A NEW
C    $         VALUE.  IF THE
C                       NEXT CHARACTER DOES NOT MATCH, USE
C    $         THE ABSOLUTE
C                       VALUE.
C                       IF ZERO, CONTINUE CHECKING FOR
C    $         MATCHES.
C                       IF POSITIVE, THIS IS THE END OF THE
C    $         STRING.
C                       USE THIS VALUE.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4  NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5  (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3  STAR, X
      DIMENSION   IA (3,270), IB (3,78), IC (3,63), ID (3,
     $ 47), IE
     1  (3,70), IF (3,12), IMP (8), NNEXT (2)
      INTEGER     OLE (8)
      EQUIVALENCE  (IA(1), IB(1)), (IA (235), IC (1)),
     $ (IA(424),
     1   ID(1)), (IA (565), IE (1)), (IA(775), IF(1))
C
C         HEADER AND DECLARATIVE STATEMENTS  (N = 1).
C
       DATA     IB / 9, 1HB, 0, 0, 1HL, 0, 0, 1HO, 0, 0,
     $ 1HC, 0, 0, 1HK,
     1   0, 0, 1HD, 0, 0, 1HA, 0, 0, 1HT, 0, 0, 1HA, 4, 8,
     $ 1HC, 0, 0,
     2   1HO, 0, 0, 1HM, 0, 0, 1HP, 0, 0, 1HL, 0, 0, 1HE,
     $ 0, 0, 1HX, 0,
     3   16, 1H , 0, 15, 1HD, 0, 0, 1HO, 0, 0, 1HU, 0, 0,
     $ 1HB, 0, 0,
     4   1HL, 0, 0, 1HE, 0, 0, 1HP, 0, 0, 1HR, 0, 0, 1HE,
     $ 0, 0, 1HC, 0,
     5   0, 1HI, 0, 0, 1HS, 0, 0, 1HI, 0, 0, 1HO, 0, 0,
     $ 1HN, 0, 8, 1HF,
     6   0, 0, 1HU, 0, 0, 1HN, 0, 0, 1HC, 0, 0, 1HT, 0, 0,
     $ 1HI, 0, 0,
     7   1HO, 0, 0, 1HN, 3, 8, 1HI, 0, 0, 1HN, 0, 0, 1HT,
     $ 0, 0, 1HE, 0.
     8   0, 1HG, 0, 0, 1HE, 0, 0, 1HR, 0,  - 15, 1H , 0, 8,
     $ 1HL, 0, 0,
     9   1HO, 0, 0, 1HG, 0, 0, 1HI, 0, 0, 1HC, 0, 0, 1HA,
     $ 0, 0, 1HL, 0,
     9   - 23, 1H , 0, 7, 1HP, 0, 0, 1HR, 0, 0, 1HO, 0, 0,
     $ 1HG, 0, 0,
     9   1HR, 0, 0, 1HA, 0, 0, 1HM, 1, 5, 1HR, 0, 0, 1HE,
     $ 0, 0, 1HA, 0,
     9   0, 1HL, 0,  - 35, 1H , 0, 0, 1HS, 0, 0, 1HU, 0, 0,
     $ 1HB, 0, 0,
     9   1HR, 0, 0, 1HO, 0, 0, 1HU, 0, 0, 1HT, 0, 0, 1HI,
     $ 0, 0, 1HN, 0,
     9   0, 1HE, 2 /
C
C         INTERNAL STATEMENTS  (N = 2).
C
       DATA     IC / 6, 1HA, 0, 0, 1HS, 0, 0, 1HS, 0, 0,
     $ 1HI, 0, 0, 1HG,
     1   0, 0, 1HN, 23, 21, 1HB, 0, 0, 1HA, 0, 0, 1HC, 0,
     $ 0, 1HK, 0, 0,
     2   1HS, 0, 0, 1HP, 0, 0, 1HA, 0, 0, 1HC, 0, 0, 1HE,
     $ 40, 0, 1HU, 0,
     3   0, 1HF, 0, 0, 1HF, 0, 0, 1HF, 0, 0, 1HR, 0, 3,
     $ 1HJ, 0, 0, 1HM,
     4   0, 0, 1H(, 30, 0, 1HO, 0, 0, 1HU, 0, 0, 1HT, 0, 0,
     $ 1H(, 31, 20,
     5   1HC, 0, 3, 1HA, 0, 0, 1HL, 0, 0, 1HL, 22, 0, 1HC,
     $ 0, 9, 1HM, 0,
     6   4, 1HM, 0, 0, 1HO, 0, 0, 1HN,  - 6, 0, 1H/, 5, 0,
     $ 1HP, 0, 0,
7     1HL, 0, 0, 1HE, 0, 0, 1HX, 9, 0, 1HN, 0, 0, 1HT,
$   0, 0, 1HI, 0,
8     0, 1HN, 0, 0, 1HU, 0, 0, 1HE, 24, 23, 1HD, 0, 3,
$   1HA, 0, 0,
9     1HT, 0, 0, 1HA, 16, 6, 1HE, 0, 0, 1HC, 0, 0, 1HC,
$   0, 0, 1HD, 0,
9     0, 1HE, 0, 0, 1H(, 32, 8, 1HI, 0, 0, 1HM, 0, 0,
$   1HE, 0, 0, 1HN,
9     0, 0, 1HS, 0, 0, 1HI, 0 /
C
      DATA     ID / 0, 1HO, 0, 0, 1HN, 7, 0, 1HO,  - 18, 0,
$   1HU, 0, 0,
1     1HB, 0, 0, 1HL, 0, 0, 1HF, 10, 33, 1HE, 0, 14,
$   1HN, 0, 5, 1HC,
2     0, 0, 1HO, 0, 0, 1HD, 0, 0, 1HE, 0, 0, 1H(, 33, 5,
$   1HD,  - 44,
3     0, 1HF, 0, 0, 1HT, 0, 0, 1HL, 0, 0, 1HE, 38, 0,
$   1HT, 0, 0, 1HF,
4     0, 0, 1HY, 35, 11, 1HO, 0, 0, 1HU, 0, 0, 1HT, 0,
$   0, 1HV, 0, 0,
5     1HA, 0, 0, 1HL, 0, 0, 1HE, 0, 0, 1HN, 0, 0, 1HC,
$   0, 0, 1HE, 0,
6     0, 1H(, 15, 0, 1HX, 0, 0, 1HT, 0, 0, 1HE, 0, 0,
$   1HF, 0, 0, 1HN,
7     0, 0, 1HA, 0, 0, 1HL, 8, 7, 1HF, 0, 0, 1HO, 0, 0,
$   1HR, 0, 0,
8     1HM, 0, 0, 1HA, 0, 0, 1HT, 0, 0, 1H(, 17 /
C
      DATA     IE / 5, 1HG, 0, 0, 1HO, 0, 0, 1HT, 0, 0,
$   1HO,  - 20, 0,
1     1H(, 19, 9, 1HI, 0, 2, 1HF, 0, 0, 1H(, 21, 0, 1HN,
$   0, 0, 1HT,
2     0, 0, 1HE, 0, 0, 1HG, 0, 0, 1HE, 0, 0, 1HR, 11, 7,
$   1HL, 0, 0,
3     1HO, 0, 0, 1HG, 0, 0, 1HI, 0, 0, 1HC, 0, 0, 1HA,
$   0, 0, 1HL, 12,
4     8, 1HN, 0, 0, 1HA, 0, 0, 1HM, 0, 0, 1HE, 0, 0,
$   1HL, 0, 0, 1HI,
5     0, 0, 1HS, 0, 0, 1HT, 43, 20, 1HF, 0, 4, 1HA, 0,
$   0, 1HU, 0, 0,
6     1HS, 0, 0, 1HE, 42, 11, 1HR, 0, 7, 1HE, 0, 0, 1HC,
$   0, 0, 1HI,
7     0, 0, 1HS, 0, 0, 1HI, 0, 0, 1HO, 0, 0, 1HN, 14, 0,
$   1HT, 0, 0,
8     1HN, 0, 0, 1HT, 27, 0, 1HU, 0, 0, 1HN, 0, 0, 1HC,
$   0, 0, 1HH,
9     29, 14, 1HP, 0, 0, 1HE, 0, 4, 1HA, 0, 2, 1HD,  -
$   26, 0, 1H(,
9     25, 0, 1HL, 13, 4, 1HT, 0, 0, 1HU, 0, 0, 1HR, 0,
$   0, 1HN, 36, 0,
      9   1HW, 0, 0, 1HI, 0, 0, 1HN, 0, 0, 1HD, 39, 4, 1HS,
      $ 0, 0, 1HT, 0,
      9   0, 1HO, 0, 0, 1HF, 34, 5, 1HT, 0, 0, 1HY, 0, 0,
      $ 1HP, 0 /
C
        DATA     IF / 0, 1HE, 0,  - 154, 1H , 0, 4, 1HU, 0,
      $ 0, 1HS, 0, 0,
      1   1HE, 0, 0, 1H(, 37, 0, 1HW, 0, 0, 1HF, 0, 0, 1HI,
      $ 0, 0, 1HT, 0,
      2   0, 1HE, 0, 0, 1H(, 28 /
        DATA     IMP / 1HI, 1HM, 1HP, 1HL, 1HI, 1HC, 1HI, 1HT
      $ /
        DATA     OLE / 1HO, 1HV, 1HE, 1HR, 1HL, 1HA, 1HY, 1H(
      $ /
C
        DATA     NNEXT / 1, 79 /
C
        ISTART     = IPOINT
        NEXT       = NNEXT(N)
        GO TO 3
    1   NEXT       = NEXT + 1
C
C           ADVANCE TO THE NEXT CHARACTER OF ISTATE
C
        IPOINT     = IPOINT + 1
    2   IF (IPOINT .GT. ICHARS) GO TO 11
    3   IF (LSTATE(IPOINT) .NE. IBLANK) GO TO 4
C
C           SUPPRESS ANY BLANKS
C
        CALL SHIFTL (IBLANK, IPOINT, ICHARS, LSTATE(1))
        GO TO 2
C
C           NOW CHECK FOR A CHARACTER MATCH.  IF ALREADY
C     $       PAST, USE THE
C           DEFAULT TERMINATION, IDENT = 45.
C
    4   IF (LSTATE(IPOINT) .LT. IA(2,NEXT)) GO TO 11
        IF (LSTATE(IPOINT) .GT. IA(2,NEXT)) GO TO 6
C
C           MATCH FOUND.  SEEK THE NEXT STEP.
C           -   SEARCH FOR POSSIBLE FURTHER ACTION.
C           0   CONTINUE.
C           +   DONE.
C
    5   IF  (IA(3,NEXT))  7, 1, 10
C
C           JUMP TO THE NEXT LEVEL.  CHECK CHARACTER.  DO NOT
C     $         ADVANCE
```

```
C           IPOINT.
C           0  DONE.
C           - OR +  JUMP TO THIS LOCATION IN IA(I,NEXT) +
C     $           NEXT.
C
    6  IF (IA(1,NEXT) .EQ. 0) GO TO 11
       NEXT        = IA(1, NEXT) + NEXT
       GO TO 4
C
C           IF NEGATIVE, THERE MAY BE ADDITIONAL CHARACTERS.
C     *          IF NOT, TAKE
C           THIS VALUE OF IA(3,NEXT).
C
    7  IDENT       =  - IA(3, NEXT)
C
C           ADVANCE TO THE NEXT CHARACTER OF ISTATE.
C
       IPOINT      = IPOINT + 1
    8  IF (IPOINT .GT. ICHARS) RETURN
       IF (LSTATE(IPOINT) .NE. IBLANK) GO TO 9
C
C           SUPPRESS ANY BLANKS.
C
       CALL SHIFTL (IBLANK, IPOINT, ICHARS, LSTATE(1))
       GO TO 8
    9  NEXT        = NEXT + 1
C
C           NOW CHECK FOR A CHARACTER MATCH.
C
       IF (LSTATE(IPOINT) .EQ. IA(2,NEXT)) GO TO 5
       RETURN
C
C           IF POSITIVE, WE ARE DONE.
C
   10  IDENT       = IA(3, NEXT)
       IPOINT      = IPOINT + 1
       RETURN
C
C           THE STATEMENT IS APPARENTLY A REPLACEMENT
C     $          STATEMENT.
C
   11  IPOINT      = ISTART
       IDENT       = 45
       IF (N .GT. 1) GO TO 15
C
C           CHECK FOR OVERLAY STATEMENT
C
       KC          = IPOINT
         DO 14 J   = 1, 8
   12    LC        = LSTATE(KC)
```

```
      IF (LC .NE. IBLANK) GO TO 13
C
C       SKIP BLANKS
C
      KC          = KC + 1
      GO TO 12
13    IF (LC .NE. OLE(J)) RETURN
C
C       IF WE REACH HERE, WE ARE MATCHING.
C
      KC          = KC + 1
14    CONTINUE
C
C       IF THE LOOP IS COMPLETED, THE STATEMENT IS AN
C    $        OVERLAY.
C       ASSIGN IT TYPE 14.
C
     IDENT      = 14
C
     RETURN
C
C       CHECK FOR IMPLICIT STATEMENT.
C
15   KC         = IFCINT
      DO 18 J   = 1, 8
16    LC          = LSTATE(KC)
      IF (LC .NE. IBLANK) GO TO 17
      KC          = KC + 1
      GO TO 16
17    IF (LC .NE. IMP(J)) RETURN
      KC          = KC + 1
18    CONTINUE
C
C       IF THE LOOP IS COMPLETED, THIS IS AN IMPLICIT
C    $        STATEMENT.
C       ASSIGN IT TYPE 46.
C
     IDENT      = 46
     IFOINT     = 9
     RETURN
     END
```

```
      SUBROUTINE  KF (NSTN)
C
C         THIS ROUTINE CATALOGS THE FORMAT STATEMENT
C    $            NUMBERS IN THE
C         ORDER OF THEIR USE IN THE ROUTINE.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NCLTS,
     4  NPUSH, NSNUMB, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      IF (NFORMN .LE. 0) GO TO 2
C
C         SEE IF THIS NUMBER IS ALREADY CATALOGED.
C
        DO 1 J    = 1, NFORMN
        IF (KFORM(J) .EQ. NSTN) RETURN
    1   CONTINUE
C
C         CATALOG AT THE END OF THE ARRAY.
C
    2 NFORMN     = NFORMN + 1
      IF (NFORMN .GT. MNFORM) GO TO 3
      KFORM(NFORMN) = NSTN
      RETURN
    3 PRINT  100, MNFORM, (LSTATE(I), I=1, LCHARS)
      RETURN
C
  100 FORMAT  (  *0THE ARRAY KFORM IS FULL.  THE NUMBER OF
     $ FORMAT STAT*
     1   *EMENTS CATALOGED EXCEEDED *, I5, * ON STATEMENT*
     $ /  / (20X,
     2   100A1) )
C
      END
      LOGICALFUNCTICN  KLIST (IF, NSTN)
C
C        THIS FUNCTION RECORDS THE VALUE AND POSITION OF
C     $       THE INTERNAL
C        STATEMENT NUMBERS.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      KLIST      = .FALSE.
      IF (NUMIN .GE. NUMMAX .OR. NUMIN .LT. 0) GO TO 1
      NUMIN      = NUMIN + 1
      INNUM(1, NUMIN) = IF
      INNUM(2, NUMIN) = NSTN
      KLIST      = .TRUE.
      RETURN
    1 PRINT  100, NUMMAX, (LSTATE(I), I=1, LCHARS)
      RETURN
C
  100 FORMAT  (  *0THE ARRAY INNUM IS FULL.  THE NUMBER OF
     $ INTERNAL ST*
     1   *ATEMENT NUMBERS EXCEEDED *, I5,  * ON STATEMENT*
     $ /  / (20X,
     2   100A1) )
C
      END
      LOGICALFUNCTION  KO (NSTN)
C
C        THIS FUNCTION CATALOGS THE LOCATION OF THE FORMAT
C     $         STATEMENT
C       NUMBERS IN THE ARRAY KFOUT.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      KO          = .FALSE.
      IF (NFOUT .LE. 0) GO TO 2
C
C        SEE IF THIS NUMBER IS ALREADY CATALOGED.
C
        DO 1 J    = 1, NFOUT
        IF (KFOUT(1,J) .EQ. NSTN) RETURN
    1   CONTINUE
C
C        CATALOG AT THE END OF THE ARRAY.
C
    2 NFOUT       = NFOUT + 1
      IF (NFOUT .GT. MNFORM) GO TO 3
      KFOUT(1, NFOUT) = NSTN
      KFOUT(2, NFOUT) = NEXT
      KFOUT(3, NFOUT) = ICHARS
      NEXT        = NEXT + (ICHARS + 9) / 10
      IF (NEXT .GT. MFOUT) GO TO 4
      KO          = .TRUE.
      RETURN
    3 PRINT  100, MNFORM, (LSTATE(I), I=1, LCHARS)
      RETURN
    4 PRINT  101, MFOUT, (LSTATE(I), I=1, LCHARS)
      NEXT        = KFOUT(2, NFOUT)
      NFOUT       = NFOUT - 1
      RETURN
C
  100 FORMAT  (  *0 THE ARRAY KFCUT IS FULL.  THE NUMBER
     $ OF FORMAT STA*
     1   *TEMENT NUMBERS STORED IN THE ARRAY KSNUM
     $ EXCEEDED*, I5,
     2   * AT STATEMENT* /  / (20X, 100A1) )
  101 FORMAT  (  *0 THE ARRAY LFCUT IS FULL.  THE NUMBER
     $ OF FORMAT STA*
     1   *TEMENT WORDS EXCEEDED*, I5,  * ON STATEMENT* /
     $ (20X, 100A1)
     2   )
C
      END
      SUBROUTINE  KU (N)
      COMMON  /SNLIST/  NS, REF (400, 3)
      INTEGER      REF
C
C        STORES REFERENCED STATEMENT LABELS IN ORDER
C    $         ENCOUNTERED.
C
      IF (NS .LT. 400) GO TO 1
C
C        STORAGE TABLE FULL
C
      PRINT *, "NO ROOM TO STORE STATEMENT LABEL ", N
      RETURN
C
C        ATTEMPT STORAGE
C        IS N ALREADY STORED?
C
    1 CALL SCANREF (N, NQ, NL, NR)
      IF (NQ .GT. 0) RETURN
C
C        N IS NOT IN REF.  STORE IT.
C
      NS          = NS + 1
      REF(NS, 1) = N
      IF (NS .EQ. 1) RETURN
      IF (NL .LE. 0) GO TO 2
      REF(NL, 2) = NS
      RETURN
    2 REF(NR, 3) = NS
      RETURN
C
      ENTRY  KSET
C
        DO 3 I     = 1, 400
          DO 3 J   = 1, 3
          REF(I, J) = 0
    3     CONTINUE
      NS          = 0
      RETURN
      END
      FUNCTION  NUMBS (ISTART, ISTOP, LIST)
C
C         THIS FUNCTION EXAMINES THE STRING 'LIST'
C    $         BEGINNING AT ISTART
C         LOOKING FOR A NUMERICAL VALUE WHICH IF FOUND IS
C    $         RETURNED AND
C         THE LOCATION SUPPRESSED.  OTHERWISE, A ZERO IS
C    $         RETURNED.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
    $ 50), IPOINT,
    1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
    $ 100), KSNUM
    2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
    $ (2000),
    3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
    $ NKFORM, NOUTS,
    4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
    $ NVALUE, STRING
    5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
    $ (10), IPUNCT
    1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
    $ MLCHARS,
    2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
    $ RETURN,
    3   STAR, X
      DIMENSION    LIST (1)
      IS           = ISTART
      NUMBS        = 0
    1 IF (IS .GT. ISTOP) RETURN
      IF (LIST(IS) .EQ. IBLANK) GO TO 4
        DO 2 I     = 1, 10
        IF (LIST(IS) .EQ. INTEGER(I)) GO TO 3
    2   CONTINUE
      RETURN
    3 NUMBS        = NUMBS * 10 + I - 1
    4 CALL SHIFTL (IBLANK, IS, ISTOP, LIST(1))
      ICHARS       = ICHARS - 1
      IF  (IDOLLAR .GT. 0) IDOLLAR = IDOLLAR - 1
      GO TO 1
      END
      SUBROUTINE  OUTPUT (LIST)
C
C        THIS ROUTINE WRITES THE WORK FILE RECORD FOR EACH
C     $         ROUTINE
C         STATEMENT.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IFUNCT
     1   (11), ICCUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, MCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION   LIST (1), LOUT (200)
        DO 1 I    = 1, 100
        LOUT(I)   = IBLANK
    1   CONTINUE
        DO 2 II   = 1, 200, 10
        I1        = 10 * II - 9
        I2        = MIN0(I1 + 99, ICHARS)
        NC        = I2 + 1 - I1
        IF (NC .LE. 0) GO TO 3
        ENCODE (NC,  100, LOUT (II))   (LIST (I), I=I1,
     $ I2)
    2   CONTINUE
    3 LWORDS    = (ICHARS + 9) / 10
      WRITE (LUSTATE)  ITYPE,LWORDS, ICHARS, ISNUM,
     $ (LOUT(I), I=1,
     1   LWORDS), NUMIN, ( (INNUM(I, J), I=1, 2), J=1,
     $ NUMIN)
      NOUTS     = NOUTS + 1
      RETURN
C
  100 FORMAT  ( 100A1 )
C
      END
      SUBROUTINE  QDIGIT (I, ILEFT, IMIN, LIST, N)
C
C        EVALUATE THE STRING OF INTEGERS BEGINNING AT
C    $        ILEFT
C        AND PROCEEDING LEFTWARD.
C
C        N IS THE VALUE OF THE STRING OF INTEGERS.
C        I IS THE POSITION IMMEDIATELY LEFT OF THE
C    $        INTEGERS.
C
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IFUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOLT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION   LIST (1)
      N           = - 1
      I           = NONR(IBLANK, IMIN, ILEFT - 1, LIST)
      IF (I .LE. IMIN) RETURN
C
C        FIRST DIGIT
C
        DO 1 J    = 1, 10
        IF (LIST(I) .EQ. INTEGER(J)) GO TO 2
    1   CONTINUE
C
C        IF THE LOOP IS COMPLETED, THERE IS NO DIGIT.
C
      RETURN
    2 I           = I - 1
C
C        IF ONE DIGIT IS FOUND, LOOK FOR MORE.
C
      N           = J - 1
      IF (I .LE. IMIN) RETURN
C
C        SECOND DIGIT
C
        DO 3 J    = 1, 10
        IF (LIST(I) .EQ. INTEGER(J)) GO TO 4
    3   CONTINUE
      RETURN
    4 I           = I - 1
      N           = N + 10 * (J - 1)
      IF (I .LE. IMIN) RETURN
C
C        THIRD DIGIT
```

```
C
         DO 5 J     = 1, 10
         IF (LIST(1) .EQ. INTEGER(J)) GO TO 6
    5    CONTINUE
      RETURN
    6 IF (J .LE. 1) RETURN
      N             = N + 100 * (J - 1)
      I             = I - 1
      RETURN
      END
      SUBROUTINE  SPACOUT
C
C     THIS ROUTINE INSERTS THE COMMON SPACINGS.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOLT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION    LIST (1)
      INTEGER      H
      EQUIVALENCE   (LIST(1), LSTATE(1))
C
C        IPUNCT  1  2  3  4  5  6  7  8  9 10 11
C                /  ,  (  )  *  $  .  =  -  +  "
C
      II          = IPOINT
C
C     I4 IS THE POSITION OF THE NEXT ).
C
      I4          =  - 1000
    1 IFLAG       = 0
    2 IF (II .GT. ICHARS) RETURN
        DO 3 J    = 1, 10
        IF (LIST(II) .EQ. INTEGER(J)) GO TO 5
    3   CONTINUE
C     /
    4 IF (LIST(II) .EQ. IPUNCT(1)) GO TO 9
C     ,
      IF (LIST(II) .EQ. IPUNCT(2)) GO TO 16
C     (
      IF (LIST(II) .EQ. IPUNCT(3)) GO TO 17
C     *
      IF (LIST(II) .EQ. IPUNCT(5)) GO TO 9
C     -
      IF (LIST(II) .EQ. IPUNCT(9)) GO TO 9
```

```
C     +
         IF (LIST(II) .EQ. IPUNCT(10)) GO TO 9
         GO TO 20
C
    5    N           = J - 1
    6    II          = II + 1
         IF (II .GT. ICHARS) RETURN
         IF (LIST(II) .EQ. IBLANK) GO TO 6
           DO 7 J    = 1, 10
           IF (LIST(II) .EQ. INTEGER(J)) GO TO 8
    7      CONTINUE
         IF (LIST(II) .NE. H) GO TO 4
         II          = II + N + 1
         GO TO 1
    8    N           = J - 1 + N * 10
         GO TO 6
    9    NBL         = 1
C       IF (ITYPE .EQ. 16)
         IF ( .NOT. (ITYPE .EQ. 16)) GO TO 11
C
C       DATA STATEMENT.  PUT A BLANK BEFORE EACH /, BUT NOT
C       AFTER CLOSING /.
C
C       IF (LIST(II+1) .NE. IPUNCT(2) .AND. LIST(II+1) .NE.
C     $          IBLANK)
         IF ( .NOT. (LIST(II+1) .NE. IPUNCT(2) .AND. LIST(II+
     $ 1) .NE.
     1    IBLANK)) GO TO 10
         CALL INSERT (IBLANK, II + 1, LCHARS, LIST(1), 1)
         NBL         = 2
C       END IF
   10    CONTINUE
         GO TO 15
C       END IF
   11    CONTINUE
C
C            LOOK FOR THE EXPONENTIATION OPERATOR (**)  IN
C     $           REPLACEMENT
C            STATEMENTS.  PUT A SPACE BEFORE THE FIRST AND
C     $           AFTER THE
C            SECOND, NONE BETWEEN.
C
         IF (ITYPE .NE. 45) GO TO 12
         IF (LIST(II+1) .NE. IPUNCT(5)) GO TO 14
         CALL INSERT (IBLANK, II + 2, LCHARS, LIST(1), 1)
         CALL INSERT (IBLANK, II, LCHARS, LIST(1), 1)
         II          = II + 4
         I4          = I4 + 2
         GO TO 21
C
```

```
C     DO NOT PUT SPACE AFTER ASTERISK IN LIST - DIRECTED IO
C     STATEMENTS (ITYPE = 25, 26, 27, 28).
C
   12  CONTINUE
         DO 13 J    = 25, 28
         IF (J .EQ. ITYPE) GO TO 15
   13    CONTINUE
   14  CONTINUE
       NBL         = 2
       CALL INSERT (IBLANK, II + 1, LCHARS, LIST(1), 1)
   15  CALL INSERT (IBLANK, II, LCHARS, LIST(1), 1)
       II          = II + NBL + 1
       I4          = I4 + NBL
       GO TO 21
C
C     INSERT BLANK AFTER COMMA.
C     EXCEPT IN IF STATEMENTS (ITYPE = 21).
C
   16  IF (ITYPE .EQ. 21) GO TO 20
       CALL INSERT (IBLANK, II + 1, LCHARS, LIST(1), 1)
       II          = II + 2
       I4          = I4 + 1
       GO TO 21
   17  IF (I4 .GT. 0) GO TO 18
       I4          = MATCH(I1, ICHARS, LIST(1))
       GO TO 19
   18  IF (II .LT. I4) GO TO 20
       I4          =  - 1000
C
C     INSERT BLANK BEFORE (.
C
   19  IF (IFLAG .EQ. 1) GO TO 20
C
C     DO NOT PUT SPACES AROUND PARENTHESES IN REPLACEMENT
C     STATEMENTS.
C
       IF (ITYPE .EQ. 45) GO TO 20
       CALL INSERT (IBLANK, II, LCHARS, LIST(1), 1)
       II          = II + 2
       I4          = I4 + 1
       GO TO 1
   20  II          = II + 1
       GO TO 1
   21  IFLAG       = 1
       GO TO 2
       END
```

```
      FUNCTION  SPRESS (I, ISTOP, LIST)
C
C        THIS ROUTINE SUPPRESSES ALL BLANKS.
C
      DIMENSION   LIST (1)
      DATA     IB / 1H  /
      SPRESS     = 0.0
    1 IF (I .GT. ISTOP) GO TO 2
      IF (LIST(I) .NE. IB) RETURN
C
C        SUPPRESS ANY STRAY BLANKS.
C
      CALL SHIFIL (IB, 1, ISTOP, LIST(1))
      GO TO 1
    2 SPRESS     = 1.0
      RETURN
      END
      SUBROUTINE  STORE (JTYPE)
C
C        THIS ROUTINE ADDS DIMENSION AND TYPED VARIABLES
C    $          OF TYPE JTYPE
C        TO THE ARRAY STRING.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
    $ 50), IPOINT,
    1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
    $ 100), KSNUM
    2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
    $ (2000),
    3   LWORDS, NAME (4), NCARDS, NEXT, NFORM1, NFOUT,
    $ NKFORM, NOUTS,
    4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUNK,
    $ NVALUE, STRING
    5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
    $ (10), IPUNCT
    1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
    $ MLCHARS,
    2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
    $ RETURN,
    3   STAR, X
      DIMENSION   ITESTN (7), ITESTN1 (7), LIST (1),
    $ NEWORD (2)
      INTEGER       STRING
      EQUIVALENCE  (II, IPOINT), (ISTOP, ICHARS), (LIST(1)
    $ , LSTATE(1))
C
C        RANGE OF THE LOCATIONS N1 THRU N.
C
      N            = 0
        DO 1 I     = 1, JTYPE
        N          = N + NUMBER(I)
  1     CONTINUE
      N1           = N - NUMBER(JTYPE) + 1
  2   I3           = ISCANL(IPUNCT(3), II, ISTOP, LIST(1))
      IS           = ISCANL(IPUNCT(2), II, ISTOP, LIST(1)) -
    $ 1
      IF  (I3 - IS - 1)  3, 4, 5
  3   CALL INSERT (IBLANK, I3, LCHARS, LIST(1), 1)
      IS           = MATCH(I3 + 1, ISTOP, LIST(1))
      GO TO 5
  4   IS           = ISTOP
  5   LENGTH       = MIN0(20, IS - II + 1)
      IF (LENGTH .LE. 0) RETURN
      NEWORD(1) = IBLANK
      NEWORD(2) = IBLANK
      ENCODE (LENGTH,  100, NEWORD (1))    (LIST (K), K=II,
     $ IS)
      DECODE (7,  100, NEWORD (1))  ITESTN
      IF (N .LT. N1) GO TO 7
C
C        SEE WHETHER THIS VARIABLE IS ALREADY PRESENT IN
C    $         STRING.
C        IF SO, DROP IT.
C
        DO 6 J     = N1, N
        IF (NEWORD(1) .EQ. STRING(1,J)) GO TO 16
    6   CONTINUE
C
C        PUSH THE STRING DOWN.
C
    7 K            = NUMK = NUMK + 1
      IF (NUMK .LE. NMAX) GO TO 8
      PRINT  101, NEWORD
      RETURN
    8 IF (K .LE. N1) GO TO 10
        DO 9 I     = 1, 2
        STRING(I, K) = STRING(I, K - 1)
    9   CONTINUE
      K            = K - 1
      IF (K .GT. N) GO TO 8
C
C        INSERT THE NEW VARIABLE DEFINITION.
C
   10 NN           = N = N + 1
      NUMBER(JTYPE) = NUMBER(JTYPE) + 1
        DO 11 I    = 1, 2
        STRING(I, N) = NEWORD(I)
   11   CONTINUE
   12 IF (NN .LE. N1) GO TO 16
      NN           = NN - 1
      DECODE (7,  100, STRING (1, NN))  ITESTN1
      LENGTH       = MIN0(LENGTH, 7)
        DO 13 I    = 1, LENGTH
        IF (ITESTN1(I) .EQ. IBLANK) GO TO 16
        IF (ITESTN(I) .EQ. IBLANK) GO TO 14
        IF  (ITESTN(I) - ITESTN1(I))  14, 13, 16
   13   CONTINUE
   14   DO 15 I    = 1, 2
        STRING(I, NN + 1) = STRING(I, NN)
        STRING(I, NN) = NEWORD(I)
   15   CONTINUE
      GO TO 12
   16 II           = IS + 2
      IF (II .LE. ISTOP) GO TO 2
      RETURN
```

```
C
  100  FORMAT  ( 100A1 )
  101  FORMAT  (  *0THE ARRAY STRING IS FULL.  THE
     $ VARIABLES BEGINNING *
     1   *WITH  *, 2A10,  * WERE CROPPED.*  )
C
       END
```

```
      INTEGERFUNCTION  TRANSF (I1, I2)
C
C        THIS ROUTINE TRANSFERS THE DATA RECORD FROM ICARD
C    $           TO ISTATE.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4  NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5  (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3  STAR, X
      TRANSF     = 0
        DO 1 I      = I1, I2
        IF (LCHARS .GE. MLCHARS) GO TO 2
        LCHARS      = LCHARS + 1
        LSTATE(LCHARS) = LCARD(I)
    1   CONTINUE
      GO TO 3
    2 PRINT  100, MLCHARS, LCARD
      TRANSF     = I
      LCHARS     = MLCHARS
    3 ICHARS     = LCHARS
      RETURN
C
  100 FORMAT  (  *0THE ARRAY *LSTATE* IS FULL.  THE NUMBER
     $ OF CHARACTE*
     1   *RS IN THE CURRENT STATEMENT EXCEEDED*, I5 /  *
     $ THE ARRAY OVER*
     2   *FLOWED ON CARD*, 4X, 80A1 )
C
      END
      OVERLAY(CLEAN,2,0)
      PROGRAM  WRITES
C
C          THIS ROUTINE CONTROLS THE WRITING OF THE OUTPUT
C    $          FILE.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4  NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5  (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3  STAR, X
      COMMON  /NUMRET/  NRET
      COMMON  /SNLIST/  NS, REF (400, 3)
      DIMENSION   KCARD (200), NRSTACK (20)
      INTEGER     C, END, H, REF, RETURN, STAR, X
      I9999     = 7
      IFLAG     = 0
      NOLDTYP   = 0
      NEUFLAG   = 0
      NPUSH     = 0
      NRET      = 99
C
C          ARRAY REF CONTAINS ALL REFERENCED STATEMENT
C    $          LABELS.
C          ARRAY KSNUM CONTAINS:
C           ROW 1: ALL DEFINED STATEMENT LABELS IN THE ORDER
C    $          ENCOUNTERED.
C           ROW 2: THEIR REPLACEMENTS (1, 2, 3,...).
C          NSTATN IS THE LENGTH OF KSNUM.
C
C          COMPARE ROW 1 OF KSNUM TO REF, DELETE ANY NOT IN
C    *          BOTH.
C
      I         = 1
    1 IF (I .GT. NSTATN) GO TO 4
      CALL SCANREF (KSNUM(1, I), NO, NL, NR)
      IF (NO .GT. 0) GO TO 3
C
C        DELETE
C
      NSTATN      = NSTATN - 1
        DO 2 J      = I, NSTATN
        KSNUM(1, J) = KSNUM(1, J + 1)
    2   CONTINUE
      I           = I - 1
    3 I           = I + 1
      GO TO 1
    4 NSTATN      = 1
        DO 5 I      = 1, 4
        IF (NSTATN .LE. NBFT) GO TO 6
        NBFT        = NBFT + 100
    5   CONTINUE
    6 CONTINUE
      REWIND LUSTATE
      IF (NOUTS .LE. 0) GO TO 36
    7 READ (LUSTATE)  NTYPE, LWORDS, IC, ISNUM,  (KCARD(I),
     $ I=1,
     1  LWORDS), NUMIN, ( (INNUM(I, J), I=1, 2), J=1,
     * NUMIN)
      IF  (EOF(LUSTATE))  34, 8
    8 NOUTS       = NOUTS - 1
        DO 9 I      = 1, 1060
        LSTATE(I) = IBLANK
    9   CONTINUE
      IF (NTYPE .EQ. 0) GO TO 29
      IFLAG       = 0
      IF (NTYPE .EQ. 46) GO TO 10
C
C        IMPLICIT.  OUTPUT BEFORE OTHER SPECIFICATION
C    *         STATEMENTS.
C
      IF (NTYPE .GE. 15 .AND. NOLDTYP .LE. 6) CALL OUTSTR
      NOLDTYP     = NTYPE
   10 LCHARS      = IC + 7
      IF (IC .GT. 100) GO TO 11
      IF (IC .LE. 0) GO TO 7
      DECODE (IC,  100, KCARD (1))    (LSTATE (I), I=8,
     * LCHARS)
C        ***
      GO TO 13
   11 II          = 1
      I1          = 8
   12 I2          = MIN0(I1 + 39, LCHARS)
      ICC         = MIN0(IC, 100)
      IF (ICC .LE. 0) GO TO 13
      DECODE (ICC,  100, KCARD (II))   (LSTATE (I), I=I1,
     * I2)
```

```
      IC          = IC - 100
      II          = II + 10
      I1          = I1 + 100
      IF (IC .GT. 0) GO TO 12
   13 LCHARS      = NCNR(IBLANK, 8, LCHARS, LSTATE(1))
      IF (NTYPE .NE. 18) GO TO 15
C
C        RECORD THE DC LOOP TERMINAL STATEMENT NUMBER.
C
      NPUSH       = NPUSH + 1
      IF (NPUSH .LE. 20) GO TO 14
      PRINT  101
      LCH         = MINO(LCHARS, 99)
      PRINT  100, IBLANK, (LSTATE(I), I=1, LCH)
      STOP
   14 CONTINUE
      NPSTACK(NPUSH) = INNUM(2, 1)
   15 IF (ISNUM .EQ. 0) GO TO 23
C
C        LOCATE ISNUM IN KSNUM(1, J), REPLACE WITH
C    $          KSNUM(2, J).
C
        DO 16 J   = MSTATN, NSTATN
        IF (KSNUM(1,J) .EQ. ISNUM) GO TO 17
   16   CONTINUE
C
C        IF LOOP IS COMPLETED, THIS LABEL IS EXCESS.  SET
C    $          TO ZERO.
C
      ISNUM       = 0
      GO TO 23
   17 MSTATN      = J
      ISNUM       = KSNUM(2, J)
C
C        LABEL THE NEW STATEMENT NUMBER.
C
      ENCODE (5,  102, L)  ISNUM
      DECODE (5,  100, L)   (LSTATE(I), I=1, 5)
      IF (NPUSH .EQ. 0) GO TO 23
C
C        CHECK FOR THE DO LOOP TERMINATION STATEMENT
C    $          NUMBER.
C
      NPU         = NPUSH
        DO 18 J   = 1, NSTATN
        IF (KSNUM(2,J) .EQ. ISNUM) GO TO 19
   18   CONTINUE
      GO TO 23
   19   DO 20 I   = 1, NPU
```

```
         IF (NPSTACK(I) .EQ. KSNUM(1,J)) GO TO 21
   20    CONTINUE
       GO TO 23
C
C         IF THIS IS A TERMINATION STATEMENT, REDUCE THE
C     *        PUSH COUNT
C         AND THE STACK.
C
   21  NPUFLAG    = NPUFLAG + 1
       NPU        = NPUSH - NPUFLAG
         DO 22 II  = I, NPU
         NPSTACK(II) = NPSTACK(II + 1)
   22    CONTINUE
       IF (NPU .GE. I) GO TO 19
   23  IF (NUMIN .LE. 0) GO TO 30
C
C         INSERT ALL REVISED INTERNAL STATEMENT NUMBERS.
C
C         KSNUM(1,J) IS THE ORIGINAL STATEMENT NUMBER.
C         KSNUM(2,J) IS THE NEW STATEMENT NUMBER.
C         INNUM(1,NUMIN) IS THE POSITION IN LSTATE OF THIS
C     *         STATEMENT
C                             NUMBER.
C         INNUM(2,NUMIN) IS THE ORIGINAL STATEMENT NUMBER.
C
         DO 24 J    = 1, NSTATN
         IF (INNUM(2,NUMIN) .EQ. KSNUM(1,J)) GO TO 26
   24    CONTINUE
         DO 25 J    = 1, NFORMN
         IF (INNUM(2,NUMIN) .EQ. KFORM(J)) GO TO 27
   25    CONTINUE
       PRINT  107, INNUM (2, NUMIN)
       CALL INSERTN (INNUM(2, NUMIN), INNUM(1, NUMIN) + 7,
     $ LCHARS,
     1   LSTATE(1), 0)
       GO TO 28
   26  CALL INSERTN (KSNUM(2, J), INNUM(1, NUMIN) + 7,
     * LCHARS,
     1   LSTATE(1), 0)
       GO TO 28
   27  CALL INSERTN (J + NBFT, INNUM(1, NUMIN) + 7, LCHARS,
     $ LSTATE(1),
     1   4)
C        ***
   28  NUMIN      = NUMIN - 1
       GO TO 23
C
C        PROCESS COMMENT STATEMENTS.
C
   29  IC         = MIN0(IC, 72)
```

```
      LCHARS     = IC
      ICOUNT(1, 3) = ICOUNT(1, 3) + 1
C
C         SKIP DOUBLE BLANK RECORDS IN SUCCESSION.
C
      IF (IFLAG .EQ. 1 .AND. IC .LE. 1) GO TO 33
      IFLAG      = 0
      IF  (IC .LE. 1) IFLAG = 1
      DECODE (IC,  100, KCARD (1))   (LSTATE (I), I=1, IC)
C
   30 IF (NOUTS .EQ. 0) GO TO 34
      IF (NTYPE .EQ. 0) GO TO 32
      IF (NTYPE .EQ. 21) CALL IFSPACE
      IF (NTYPE .EQ. 18) CALL ALIGN
C
C         PUSH THE STATEMENT OVER AS REQUIRED,
C
      IF (NPUSH .LE. 0) GO TO 32
      MANY       = MIN0(2 * NPUSH, 10)
        DO 31 I    = 1, MANY
        CALL SHIFTR (IBLANK, 8, LCHARS, LSTATE(1))
   31   CONTINUE
   32 CALL PUNCHIT (NTYPE)
      ISNUM      = 0
   33 NPUSH      = NPUSH - NPUFLAG
      NPUFLAG    = 0
      IF (NOUTS .GT. 0) GO TO 7
   34 CALL PUNCHIT (NTYPE)
        DO 35 I    = 1, LCHARS
        LSTATE(I) = IBLANK
   35   CONTINUE
      CALL OUTERM
      CALL INSERT (END, 1, LCHARS, LSTATE(1), 3)
      CALL INSERT (IBLANK, 1, LCHARS, LSTATE(1), 7)
      CALL PUNCHIT (100)
      REWIND LUSTATE
   36 I9999      = 0
C
  100 FORMAT  ( 100A1 )
  101 FORMAT  (  *0  THE ARRAY NESTACK IN PROGRAM WRITES*,
     *  * OVERFLOW*
     1   *ED ON THE FOLLOWING STATEMENT.*  )
  102 FORMAT  ( I5 )
  103 FORMAT  (  *0STATEMENT NUMBER *, I6,  * WAS
     * REFERENCED BUT NOT D*
     1   *EFINED.  THE ORIGINAL VALUE WAS LEFT.*  )
C
      END
      SUBROUTINE  BRECH (LOUT, L, NN, NNN, NB, ITY)
C
C        BREAKS OUTPUT STRING 'LOUT' FOR READABILITY.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, MCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION   LOUT (L)
      IF (LCHARS .LE. IPOINT) RETURN
      IF (ITY .EQ. 16) GO TO 1
      NNN        = 10
      IF (LSTATE(IPOINT+1) .EQ. IBLANK .OR. LOUT(72) .EQ.
     $ IBLANK)
     1   RETURN
C
C        FIND THE LAST BLANK FOR THE BREAK LOCATION.
C
      N             = ISCANR(IBLANK, 62, 72, LOUT(1))
      IF (N .GE. 62 .AND. N .LE. 72) GO TO 2
      IF (ITY .NE. 17) RETURN
C
C        BREAK A FORMAT OR DATA STATEMENT ONLY AFTER A
C    $          COMMA, /, OR ).
C
    1 N2            = ISCANR(IPUNCT(2), 61, 72, LOUT(1)) + 1
      N1            = ISCANR(IPUNCT(1), 61, 72, LOUT(1)) + 1
      N4            = ISCANR(IPUNCT(4), 61, 72, LOUT(1)) + 1
      N             = MAX0(N1, N2, N4)
      IF (N .EQ. 73) GO TO 6
      NNN        = 7
      NN         = 7
      IF (N .LT. 62) GO TO 4
      NNN        = 10
      NN         = 10
      2    DO 3 I      = N, 72
           LOUT(I)     = IBLANK
      3    CONTINUE
C
C            FOR FIRST OR SINGLE CARD, IPOINT = 72
C
           IPOINT       = N + IPOINT - 72
           RETURN
      4    IF (ITY .EQ. 16) RETURN
C
C            HERE TO INSERT AN ASTERISK INTO A FORMAT
C     $               STATEMENT.
C
           IF   (NB .LT. 0) NB = NN
           N5            = ISCANR(IPUNCT(5), NB, 71, LOUT(1))
           N11           = ISCANR(IPUNCT(11), NB, 71, LOUT(1))
           J             = 5
           IF (N11 .LT. N5) GO TO 5
           N5            = N11
           J             = 11
      5    NH            = ISCANR(H, NB, 71, LOUT(1))
           IF (NH .GT. N5 .OR. N5 .LT. NB) RETURN
           LOUT(72)     = IPUNCT(J)
           IPOINT       = IPOINT - 1
           LSTATE(IPOINT) = IPUNCT(J)
           IPOINT       = IPOINT - 1
      6    NNN          = 10
           NN           = 10
           RETURN
           END
      SUBROUTINE  FIXDATA
C
C         THIS SUBROUTINE ASSURES THAT THE HOLLERITH FIELDS
C     $        IN DATA
C         STATEMENTS ARE PROPERLY HANDLED.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1  IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2  (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3  LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4  NPUSH, NSNUMS, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5  (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IFUNCT
     1  (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2  MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3  STAR, X
C
C         SCAN FOR THE H WHICH MAY BE THE START OF A
C     $        HOLLERITH FIELD.
C
      INTEGER      H
      I1           = 10
    1 IH           = ISCANL(H, I1, LCHARS, LSTATE(1))
      IF (IH .GE. LCHARS) RETURN
      IS           = IH - 1
C
C         DETERMINE WHETHER THE H IS PRECEDED BY AN
C     $        INTEGER.
C
      IF (LSTATE(IS) .EQ. IBLANK) GO TO 9
        DO 2 I     = 1, 10
        IF (LSTATE(IS) .EQ. INTEGER(I)) GO TO 3
    2   CONTINUE
      GO TO 9
    3 N            = I - 1
      IS           = IS - 1
      IF (LSTATE(IS) .EQ. IBLANK) GO TO 6
        DO 4 I     = 1, 10
        IF (LSTATE(IS) .EQ. INTEGER(I)) GO TO 5
    4   CONTINUE
      GO TO 9
    5 N             = N + 10 * (I - 1)
      IS            = IS - 1
      IF (LSTATE(IS) .EQ. IBLANK) GO TO 8
        DO 6 I      = 1, 10
        IF (LSTATE(IS) .EQ. INTEGER(I)) GO TO 7
    6   CONTINUE
      GO TO 9
    7 N             = N + 100 * (I - 1)
      IS            = IS - 1
      IF (LSTATE(IS) .NE. IBLANK) GO TO 9
C
C         DETERMINE IF THE INTEGER IS PRECEDED BY A /,
C    $          COMMA, OR *.
C
    8 IS            = IS - 1
      IF (LSTATE(IS) .EQ. IPUNCT(1)) GO TO 10
      IF (LSTATE(IS) .EQ. IPUNCT(2)) GO TO 10
      IF (LSTATE(IS) .EQ. IPUNCT(5)) GO TO 10
    9 I1            = IH + 2
      GO TO 1
   10 IS            = IH + N
      IH            = IH + 1
        DO 11 I     = IH, IS
        LSTATE(I) = LSTATE(I) + 1
   11   CONTINUE
      I1            = IS + 3
      GO TO 1
      END
      SUBROUTINE  IFSPACE
C
C          THIS SUBROUTINE COMPLETES THE SPACING WITHIN IF
C    $          STATEMENTS.
C
      COMMON /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
C
C          FIND THE FIRST ( (= IPUNCT(3)).
C
      LOGICAL      CHECK
      LLOWER       = ISCANL(IPUNCT(3), 11, LCHARS, LSTATE(1))
C
C          FIND THE MATCHING ) (=IPUNCT(4)).
C
      LUPPER       = MATCH(LLOWER, LCHARS, LSTATE(1))
C
C          FIND THE FIRST . (= IPUNCT(7))
C
      IPFIRST      = ISCANL(IPUNCT(7), LLOWER + 1, LCHARS,
     $ LSTATE(1))
      IF (IPFIRST .GE. LUPPER) RETURN
C
C          FIND THE NEXT .
C
    1 IPNEXT       = ISCANL(IPUNCT(7), IPFIRST + 1, LCHARS,
     $ LSTATE(1))
      IF (IPFIRST .GE. LUPPER) RETURN
      IF (IPNEXT-IPFIRST .GT. 4) GO TO 9
      IF  (IPNEXT - IPFIRST - 3)  9, 2, 7
C
C          TWO CHARACTERS.  ARE THEY EQ, GE, GT, LE, LT, NE,
C    $          OR?
```

```
C
    2  IF (LSTATE(IPFIRST+1) .EQ. 1HE) GO TO 3
       IF (LSTATE(IPFIRST+1) .EQ. 1HG) GO TO 4
       IF (LSTATE(IPFIRST+1) .EQ. 1HL) GO TO 4
       IF (LSTATE(IPFIRST+1) .EQ. 1HN) GO TO 5
       IF (LSTATE(IPFIRST+1) .EQ. 1HO) GO TO 6
       GO TO 9
C
    3  IF (LSTATE(IPFIRST+2) .EQ. 1HQ) GO TO 8
       GO TO 9
C
    4  IF (LSTATE(IPFIRST+2) .EQ. 1HT) GO TO 8
    5  IF (LSTATE(IPFIRST+2) .EQ. 1HE) GO TO 8
       GO TO 9
C
    6  IF (LSTATE(IPFIRST+2) .EQ. 1HR) GO TO 8
       GO TO 9
C
C          THREE CHARACTERS.  ARE THEY AND OR NOT?
C
    7  IF (CHECK(3HAND,3,IPFIRST+1,IPNEXT,LSTATE(1),IP)) GO
     $ TO 8
       IF ( .NOT. CHECK(3HNOT,3,IPFIRST+1,IPNEXT,LSTATE(1),
     $ IP)) GO TO 9
C
C          YES.  INSERT SURROUNDING SPACES.
C
    8  CALL INSERT (IBLANK, IPNEXT + 1, LCHARS, LSTATE(1),
     $ 1)
       CALL INSERT (IBLANK, IPFIRST, LCHARS, LSTATE(1), 1)
       IPNEXT     = IPNEXT + 2
    9  IPFIRST    = IPNEXT
       GO TO 1
       END
```

```
      SUBROUTINE  OUTFRM
C
C        THIS SUBROUTINE OUTPUTS THE FORMAT STATEMENTS IN
C    $        THE ORDER
C        THEY ARE USED.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
    $ 50), IPOINT,
    1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
    $ 100), KSNUM
    2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
    $ (2000),
    3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
    $ NKFORM, NOLTS,
    4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
    $ NVALUE, STRING
    5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
    $ (10), IPUNCT
    1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
    $ MLCHARS,
    2   MNFORM, MNSTATE, MCARD, NMAX, NUMMAX, PROGRAM (7),
    $ RETURN,
    3   STAR, X
      COMMON  /NUMFMT/  NRFT
      INTEGER        A1, C, FORMAT, H
      DATA    A1, FORMAT / 2HA1, 6HFORMAT /
      IF (NFORMN .LE. 0 .OR. IERROR .EQ. 999) RETURN
      LCHARS     = 1
      LSTATE(1) = C
      CALL PUNCHIT (0)
      IERROR     = 999
C
C        KFOUT(1,JJ) IS THE ORIGINAL FORMAT STATEMENT
C    $        LABEL.
C        KFOUT(2,JJ) IS THE FIRST WORD THIS FORMAT
C    $        OCCUPIES IN LFOUT.
C        KFOUT(3,JJ) IS THE NUMBER OF CHARACTERS IN THIS
C    $        FORMAT.
C
C        MAIN LOOP
C
        DO 25 J    = 1, NFORMN
         DO 1 JJ    = 1, NFOUT
          IF (KFORM(J) .EQ. KFOUT(1,JJ)) GO TO 2
    1     CONTINUE
C
C        NOT FOUND.  INSERT A DUMMY FORMAT STATEMENT.
C
```

```
      PRINT  100, KFORM (J)
      LCHARS    = ICHARS = 0
      CALL INSERTS (IPUNCT(4), 1, LCHARS, LSTATE(1), 1)
      CALL INSERTS (A1, 1, LCHARS, LSTATE(1), 3)
      CALL INSERTS (IPUNCT(3), 1, LCHARS, LSTATE(1), 2)
      GO TO 7
C
C      RETRIEVE THE FORMAT STATEMENT FROM THE ARRAY
C   $     KFOUT.
C
 2    IN        = KFOUT(2, JJ)
      LCHARS    = ICHARS = KFOUT(3, JJ)
      NFOUT     = NFOUT - 1
        DO 4 I    = 1, 3
        IF (NFOUT .LT. JJ) GO TO 4
          DO 3 JJJ  = JJ, NFOUT
          KFOUT(I, JJJ) = KFOUT(I, JJJ + 1)
 3        CONTINUE
        KFOUT(I, NFOUT + 1) = 0
 4      CONTINUE
      IPOINT    = 1
        DO 5 II   = IN, 1000, 10
        I2        = MIN0(IPOINT + 99, ICHARS)
        IC        = I2 + 1 - IPOINT
        IF (IC .LE. 0) GO TO 6
        DECODE (IC,  101, LFOUT (II))    (LSTATE (I),
   $ I=IPOINT, I2)
        IPOINT    = IPOINT + 100
 5      CONTINUE
C
C      COMPLETE THE FORMAT STATEMENT.
C
 6    CALL INSERTS (IBLANK, ICHARS + 1, LCHARS,
   $ LSTATE(1), 1)
      CALL INSERTS (IPUNCT(4), ICHARS + 1, LCHARS,
   $ LSTATE(1), 1)
      CALL INSERTS (IPUNCT(3), 1, LCHARS, LSTATE(1), 2)
 7    CALL INSERTS (FORMAT, 1, LCHARS, LSTATE(1), 8)
      CALL INSERTS (IBLANK, 1, LCHARS, LSTATE(1), 2)
      CALL INSERTN (J + NBFT, 1, LCHARS, LSTATE(1), 4)
C      ***
      II        = 18
C
C      SPACE OUT THE BALANCE OF THE FORMAT STATEMENT.
C
 8    IF (II .GE. LCHARS) GO TO 25
      IS        = II
C
C      SEARCH FOR THE FIRST SPECIAL CHARACTER.
C
```

```
      DO 10 II   = IS, LCHARS
C       SLASH
        IF (LSTATE(II) .EQ. IFUNCT(1)) GO TO 21
C       COMMA
        IF (LSTATE(II) .EQ. IFUNCT(2)) GO TO 23
C       ASTERISK OR DOUBLE QUOTE
          DO 9 JJ    = 5, 11, 6
          IF (LSTATE(II) .EQ. IFUNCT(JJ)) GO TO 18
    9     CONTINUE
        IF (LSTATE(II) .EQ. H) GO TO 11
   10   CONTINUE
      GO TO 25
C
C       H DETECTED.  IS THIS A HOLLERITH FIELD?
C
   11 IPR         = II - 2
        DO 12 I   = 1, 10
        IF (INTEGER(I) .EQ. LSTATE(II-1)) GO TO 13
   12   CONTINUE
C
C       NO.  REPEAT THE SEARCH.
C
      II          = II + 1
      GO TO 8
C
C       HOLLERITH FIELD.  DETERMINE ITS LENGTH.
C
   13 N           = I - 1
        DO 14 I   = 1, 10
        IF (INTEGER(I) .EQ. LSTATE(II-2)) GO TO 15
   14   CONTINUE
      GO TO 16
   15 N           = N + 10 * (I - 1)
      IPR         = IPR - 1
      IF  (INTEGER(2) .EQ. LSTATE(II - 3)) N = N + 100
      IF  (N .GE. 100) IPR = IPR - 1
   16 IF (LSTATE(IPR) .EQ. IBLANK) GO TO 17
      CALL INSERTS (IBLANK, IPR, LCHARS, LSTATE(1), 1)
      II          = II + 1
   17 ILAST       = II + N
      IFIRST      = II + 1
      GO TO 19
C
C       INSERT A BLANK BEFORE AN * OR ", THEN SKIP TO THE
C   $         NEXT * OR ".
C
   18 CALL INSERTS (IBLANK, II, LCHARS, LSTATE(1), 1)
      IFIRST      = II + 2
      ILAST       = ISCANL(IFUNCT(JJ), IFIRST, LCHARS,
     * LSTATE(1))
   19    II          = ILAST + 1
C
C         ALTER HOLLERITH FIELDS TO ASSURE PROPER OUTPUT
C    $         SPACING.
C
          DO 20 I    = IFIRST, ILAST
          LSTATE(I) = LSTATE(I) + 1
   20     CONTINUE
         IF (II .GE. LCHARS) GO TO 25
         IF (LSTATE(II) .EQ. IFUNCT(1)) GO TO 21
         IF  (LSTATE(II) .EQ. IFUNCT(2)) II = II + 1
         GO TO 24
C
C         INSERT A BLANK BEFORE THE FIRST AND AFTER THE
C    $         LAST /.
C
   21    CALL INSERTS (IBLANK, II, LCHARS, LSTATE(1), 1)
         II          = II + 2
   22    IF (LSTATE(I) .NE. IFUNCT(1)) GO TO 24
         II          = II + 1
         GO TO 22
C
C         INSERT A BLANK AFTER A COMMA.
C
   23    II          = II + 1
C
C         INSERT A BLANK.
C
   24    CALL INSERTS (IBLANK, II, LCHARS, LSTATE(1), 1)
         II          = II + 1
         GO TO 8
   25    CALL PUNCHIT (17)
C
C         END MAIN LOOP
C
        LCHARS     = 1
        LSTATE(1) = 0
        CALL PUNCHIT (0)
        RETURN
C
  100   FORMAT  (  *0 COULD NOT FIND FORMAT NUMBER *, I5,  *
     $ IN THE AREA*
     1   *Y KEOUT.  A DUMMY FORMAT STATEMENT (A1) WAS
     $ INSERTED.*  )
  101   FORMAT  ( 100A1 )
C
        END
```

```
      SUBROUTINE OUTSID
C
C        THIS ROUTINE SETS UP THE FINAL TYPE AND DIMENSION
C     $           STATEMENT
C        RECORDS.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (3,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IFUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MICHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION    KTYPE (7)
      INTEGER      STRING, TEST (20)
      DATA     KTYPE / 9HDIMENSION, 8HEXTERNAL, 7HCOMPLEX,
     $ 6HDOUBLE,
     1   7HINTEGER, 7HLOGICAL, 4HREAL /
      NE           = 0
        DO 3 J     = 1, 7
C
C         SKIP THE TYPE IF NONE OCCUR.
C
        IF (NUMBER(J) .EQ. 0) GO TO 3
C
C         INSERT THE TYPE NAME.
C
        N            = NUMBER(J)
        LCHARS       = 7
        CALL INSERTS (KTYPE(J), 8, LCHARS, LSTATE(1), 10)
        LCHARS       = 19
        IPOINT       = 20
C
C         INSERT ONE VARIABLE AT A TIME.
C
          DO 1 K     = 1, N
          NE         = NE + 1
          DECODE (20,  100, STRING (1, NE))  TEST
```

```
            NN          = NONR(IBLANK, 1, 20, TEST(1))
            CALL INSERTS (STRING(1, NE), IPOINT, LCHARS,
     $ LSTATE(1), NN)
            IPOINT      = LCHARS + 1
            IF (K .EQ. N) GO TO 2
            CALL INSERTS (IPUNCT(2), IPOINT, LCHARS,
     $ LSTATE(1), 2)
            IPOINT      = LCHARS + 1
    1       CONTINUE
    2     CALL PUNCHIT (J + 6)
          NUMBER(J) = 0
    3     CONTINUE
        RETURN
C
  100   FORMAT   ( 100A1 )
C
        END
```

```
      SUBROUTINE  PUNCHIT (ITY)
C
C         THIS ROUTINE WRITES THE REORGANIZED STATEMENTS ON
C    $          THE OUTPUT
C         FILE TAPE4.  THIS FILE IS READY FOR COMPILATION
C    $          OR PUNCHING.
C
      COMMON  /ALL/  ICHARS, IDOLLAR, IERROR, INNUM (2,
     $ 50), IPOINT,
     1   IPROG, ISNUM, ITYPE, I9999, KFORM (100), KFOUT (7,
     $ 100), KSNUM
     2   (2, 400), LCARD (80), LCHARS, LFOUT (1000), LSTATE
     $ (2000),
     3   LWORDS, NAME (4), NCARDS, NEXT, NFORMN, NFOUT,
     $ NKFORM, NOUTS,
     4   NPUSH, NSNUMC, NSTATN, NUMBER (7), NUMIN, NUMK,
     $ NVALUE, STRING
     5   (2, 100)
      COMMON  /DATA/  C, END, H, IBLANK, IEOF, INTEGER
     $ (10), IPUNCT
     1   (11), ICOUNT (2, 4), LUIN, LUOUT, LUSTATE, MFOUT,
     $ MLCHARS,
     2   MNFORM, MNSTATE, NCARD, NMAX, NUMMAX, PROGRAM (7),
     $ RETURN,
     3   STAR, X
      DIMENSION    LOUT (72)
      INTEGER      C, H, STAR, X
      IF (ITY .EQ. 16) CALL FIXDATA
      IPOINT      = 72
      NNN         = 7
      NB          = 16
        DO 1 I      = 1, 72
        LOUT(I)     = LSTATE(I)
    1   CONTINUE
C
C          ONE OF THE FIRST CARD OUTPUT.
C
      CALL BRECH (LOUT, 72, NN, NNN, NB, ITY)
      WRITE (LUOUT,   100)  LOUT
      IF (LCHARS .LE. 72) GO TO 7
C
C         MULTIPLE CARD OUTPUT.
C         INDENT THE REMAINDER.
C         NN IS THE STARTING LOCATION FOR THE CONTINUATION
C    $          CARDS.
C
      IC          = 1
      NN          = 10 + 2 * NPUSH
C
```

```
C          FORMAT OR DATA STATEMENT.  CANNOT BE INDENTED.
C
      IF  (ITY .EQ. 16 .OR. ITY .EQ. 17) NN = NNN
    2   DO 3 I     = 1, 72
        LOUT(I)    = IBLANK
    3   CONTINUE
    4 IF (LSTATE(IPOINT+1) .NE. IBLANK) GO TO 5
      IPOINT     = IPOINT + 1
      IF (IPOINT .GE. LCHARS) GO TO 5
      GO TO 4
    5   DO 6 I     = NN, 72
        IPOINT     = IPOINT + 1
        LOUT(I)    = LSTATE(IPOINT)
    6   CONTINUE
      IC         = MIN0(IC + 1, 10)
      LOUT(6)    = INTEGER(IC)
      NE         =  - 1
      CALL BRECH (LOUT, 72, NN, NNN, NE, ITY)
      WRITE (LUOUT,  100) LOUT
      IF (IPOINT .LT. LCHARS) GO TO 2
C
    7   DO 8 I     = 1, LCHARS
        LSTATE(I) = IBLANK
    8   CONTINUE
      LCHARS     = 0
      RETURN
C
  100 FORMAT  ( 76A1 )
C
      END
```

APPENDIX B

Function and Subroutine Descriptions

Only those routines which have been added or significantly modified are discussed here. See the original report for descriptions of the remainder.

ROUTINE	DESCRIPTION
ALIGN	A subroutine to position the equals sign in replacement and DO statements. It calls INSERT and ISCANL. It is called from READS and WRITES.
BLANKS	A subroutine to delete blanks from a statement. It also decides whether a dollar sign is a statement separator or a Hollerith character. Hollerith strings are detected and modified to preserve any blanks in them. It is called from READS and calls DOLLAR, ISCANL, NONR, QDIGIT, and SPRESS.
BRECH	A subroutine called from WRITES to break a statement into one-line increments. The logic in BRECH was originally in two places in WRITES. BRECH calls ISCANR.
DOLLAR	A logical function called from BLANKS to determine whether a dollar sign is a statement separator or a character. If the $ is a separator, DOLLAR returns the value .TRUE. It calls IDENT, ISCANL, MATCH, NONL, and NONR.
IDENT	An integer function called from READS and DOLLAR to determine the type of statement being processed. In returns an integer code identifying the statement type. Data has been added to identify OVERLAY (IDENT = 14) and IMPLICIT (IDENT = 46) statements. IDENT calls SHIFTL.

KU	A subroutine called by READS to compile a list of referenced statement labels. The list is stored as a binary tree in array REF, with the count in NS. Entry KSET sets REF and NS to 0. KU calls SCANREF.
QDIGIT	A subroutine called by BLANKS to evaluate the character count in front of a Hollerith string, e.g., 6HSTRING. If the character before the H is not a digit, QDIGIT returns -1. QDIGIT calls NONR.
SCANREF	A subroutine called by KU and WRITES to see whether the statement label N has been stored in the binary tree REF. If N is in REF, its subscript is returned in NQ. Else, the next left pointer is returned in NL or the next right pointer is returned in NR, depending on where N fits in the tree.
WRITES	The main program in the output overlay. It writes the reorganized routine to TAPE 4, which is equated to TFILE in the main program, CLEAN. WRITES deletes unreferenced statement labels from each routine as follows: READS stores each defined statement label in the array KSNUM and each referenced label in the array REF. The variable NSTATN counts the entries in KSNUM. READS writes a complete statement including its original label to TAPE 3. WRITES compares the list of defined labels to the list of referenced labels and deletes any which are not in both lists. The logic looks like this:

```
          I = 1
          WHILE (I .LE. NSTATN)
C           NSTATN IS THE LENGTH OF KSNUM
            CALL SCANREF (KSNUM(I), NQ, NL, NR)
            IF (NQ .LE. 0)
C              I.E. KSNUM(I) NOT IN REF
```

```
            NSTATN = NSTATN - 1
            DO (J = 1, NSTATN)
              KSNUM(J) = KSNUM(J + 1)
            END DO
            I = I - 1
C             LOCATION I HAS A NEW VALUE, MUST BE CHECKED
C             AGAIN
        END IF
        I = I + 1
    END WHILE
```

Then, as each statement is read from TAPE 3, its label, if any, is checked against KSNUM. If a match is found, the subscript from KSNUM is used as the new label, since the entries in KSNUM are stored in the order encountered. If no match is found, the label is redundant and no label is output.

*U.S.Government Printing Office: 1979 — 657-002/676